AN ABSTRACT OF THE THESIS OF

Chih–Ming Chang for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on January 24, 1997.

Title: Performance Modeling and Analysis of Asynchronous Pipelines for Designers

Redacted for Privacy

Abstract approved: _____
Shih–Lien Lu

Better performance has been one of the main motivations behind the recent resurgence of interest in asynchronous circuits (no matter whether this is always true or not). We are particularly interested in the performance of pipelines since they are used extensively in current digital systems. There exists an algorithm that can find the exact upper and lower bounds on the separation time of events in a certain class of process graphs. However, some transformations and complex mathematical analyses, such as graph decomposition for infinite unfolded process graphs must be employed in order to reach exact bounds. This algorithm may be a good candidate for the application of CAD tool development and circuit synthesis, but it tends to block designers from visualizing what factors really affect the performance of asynchronous circuits.

In this thesis, a simple approach is adopted to approximate the performance bounds. Since our method is a symbolic approach instead of a numerical approach, it allows designers to analyze the circuit performance while providing design guidelines and approaches at the same time. Our approach has two steps. First, several basic modules are chosen, including FIFO, Fork, Join, Toggle/XOR, Arbiter/Call and Select/

XOR. The individual output loop delay, equivalent input delay and equivalent output delay are derived based on the Equal loop–delay theorem. The result is a set of difference equations. The performance approximation can be obtained with simple mathematical operations on the difference equations, given the bounds of stage–delays. That is, the performance bounds of output loop delay, equivalent input delay and equivalent output delay can be represented as the bounds of stage–delays. Second, for a larger system consisting of those basic modules, its performance bounds can be derived directly from the bounds of output loop delay, equivalent input delay and equivalent output delay of those basic modules which have been obtained already. This approach allows a fast and easy calculation of performance bounds, avoiding the need to re–derive the difference equations for the whole system. Both modular design and performance approximation are possible with our approach.

Performance Modeling and Analysis of Asynchronous Pipelines for Designers

by

Chih–Ming Chang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed January 24, 1997
Commencement June, 1997

Doctor of Philosophy thesis of Chih–Ming Chang presented on January 24, 1997

APPROVED:

Redacted for Privacy

Major Professor, representing Electrical and Computer Engineering

Redacted for Privacy

Head of Department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University Libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Chih–Ming Chang, Author

ACKNOWLEDGMENT

I am grateful to my advisor, Professor Shih–Lien Lu, and the rest of the committee members for their guidance. This research would never have been completed without Professor Lu's continuous support in both finances and spirit. He provided technical consultation and possible direction toward the answers whenever I encountered problems. His patience and encouragement gave me full confidence in doing the project.

This research would not be complete without obtaining the exact performance bounds. I am grateful to Professor Henrik Hulgaard in the Department of Information Technology, Technical University of Denmark. He provided me with the simulation tool and taught me how to use it. His quick responses to my questions enabled me to complete this research in a timely way.

My family's continuous support for my 25 years of student life is highly appreciated. Their understanding and consideration relieved me from all other pressures while pursuing this advanced degree.

This work was supported by NSF grant #MIP–9211510.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF APPENDIX FIGURES

PERFORMANCE MODELING AND ANALYSIS OF ASYNCHRONOUS PIPELINES FOR DESIGNERS

1. INTRODUCTION

We all live in a clock world. In daily life our living habits — when to wake up, to start working, to have a meeting, to call it a day, to go to bed — are more or less regulated by the clock. The same type of phenomenon occurs in the world of digital circuits, although the preciseness in clock regulation is different. We can wake up a little bit earlier or later in our daily lives, but with digital circuits, which are constructed using clocked logic, the data must be stable before the clock arrives. Can we imagine what our lives would be like without clocks to regulate our living tempo? Will our lives or the whole society become more efficient? It is not the intention of this thesis to answer these particular questions; however, we are interested in the related question — what will happen if digital circuits are designed without clocks? In this chapter, we will discuss the advantages and disadvantages of leaving the clock out of digital circuits. Our major interest is how the lack of a clock affect the performance of a digital circuit. The goal of this thesis and the approach taken to fulfill this goal will also be presented in this chapter.

1.1 Motivation

Clocked–logic design is now the most common discipline being used in the design of digital systems. The main reason for this design discipline being so prevalent in the industry and in academies is because the knowledge required to design efficient and effective circuits using this discipline is simple and easy for most design engineers. Combinational logic circuits and registers are the two parts comprised in a digital system. Boolean algebra is the only basic tool necessary to design a combinational logic circuit.

Moreover, the key discipline in designing a correct and functional circuit (e.g., state machine) is to stabilize the data before they are latched into registers by the clock. The tools and discipline themselves do not change, but the way for not violating the discipline is getting hard to observe nowadays. This is true because, as the die size grows and transistor feature sizes continue to shrink, wiring delays play a more and more important role compared with gate delays when a high–speed digital system is considered (implying that a higher frequency clock is used). That is, when we go down to layout level, the wiring delays can no longer be ignored and can possibly skew the timing of the whole system. The wiring delays may not affect the data processing too much since most of the data communicate locally. This is, however, not the case for the clock signal. The clock is usually generated in a certain place and then distributed globally to the whole chip. To make the whole system function correctly, clock skew (phase delay due to long wiring) should be controlled so that the design discipline is not violated. Due to its global distribution, controlling clock skew is not a trivial matter. Not to mention the appropriate distribution of the clock, the clock itself may take up a lot of the die's area. For example, more than a quarter of the silicon for the Alpha chip from Digital Equipment Corporation (DEC) [1] is devoted to clock logic. The potential problems resulting from the clock have triggered researchers to re–think the feasibility of eliminating the clock from the design of a digital system.

The way to approach the design of a clocked system is usually called synchronous design methodology. This is because, in this kind of system, the processing data must always be synchronized with the clock. The design methodology which abandons the usage of the clock in designing a digital system is called asynchronous (self–timed) design methodology [2]. The design discipline in asynchronous design methodology is to set up a communication protocol (handshaking) between communicating parties. The mechanism for implementing this protocol is served as the function of synchronization,

which ensures that the data have been latched correctly, and is performed locally in asynchronous digital systems.

Asynchronous circuits are said to have several advantages over synchronous circuits [1,3,4].

(1) *Clock skew free*: Clock skew is not only hard to control but also degrades overall performance compared with ideal non–clock skew systems [5]. Since asynchronous circuits use a local communication scheme (handshaking) between successive stages and feature no global clock, clock skew can be avoided naturally.

(2) *Noise reduction*: In a conventional clock–driven system, all the activating transistors in the system switch at the same time when the global clock arrives. This leads to steep source current variation and results in high–voltage inductive noise in the power line. In a self–timed system, handshaking is distributed with time, dramatically reducing the number transistors switching simultaneously. This reduces power noise.

(3) *Zero stand–by power*: If made with a Complementary Metal–Oxide Semiconductor (CMOS), then , theoretically, a self–timed system consumes no power (in reality, a small amount of leakage current exists) when it is idling (not undertaking any handshaking). On the contrary, a clock–driven system consumes power even when no tasks are being executed, since the clock continues to operate. If an asynchronous microprocessor is running under full (maximum) load (speed), the power consumption will be compatible with a synchronous microprocessor. In reality, however, the microprocessor is rarely running at the maximum speed in general applications, leading to less total power consumption. This characteristic is especially useful for portable devices.

(4) *Low heat generation*: More power is consumed implying more heat is generated. In Very Large–Scaled Integrated circuit (VLSI) design, when the density is getting higher and higher or more specific the size reduces to 0.5 micro level and below, tens of millions to hundreds of millions of transistors could be contained in a single chip. With this kind of density, the heat dissipation is a serious problem. Although a static

CMOS is employed to save the power, a clocked microprocessor still intrinsically wastes the power and therefore dissipates the "extra" heat. As mentioned before, asynchronous microprocessors consume less power, hence, less heat is generated.

(5) *Modularity and composibility*: The feature of handshaking as a means of synchronization within asynchronous systems enables the hierarchical design of self–timed circuits. A larger module can be constructed easily and efficiently by appropriately interconnecting smaller modules using a self–timed signaling protocol. This larger module will function correctly as long as the smaller modules are correctly designed. No additional attention needs to be paid to timing. In conventional clocked circuit design, composing two modules directly is, in general, disallowed since this might destroy the synchronization due to additional and unexpected delays.

(6) *Performance improvement*: In conventional clocked circuit design (e.g., clocked pipeline structure), the clock period must be greater than the worst stage–delay in order not to override some data. That is, the overall performance is bounded by the worst–case delay. In self–timed circuit design, the current stage result can be transmitted to the next stage as long as the next stage is ready to accept data. From the overall viewpoint, only average delay is seen instead of worst–case delay. This will speed up the throughput. As technology improves, due to the characteristics of modularity and composibility, some old–version module(s) can be replaced by new and fast version module(s) without redesigning the whole system. This improves the performance of the whole system and, hence, increases the product's market life with the least price.

Although there are some potential benefits for asynchronous circuits over synchronous circuits, as stated above, the design of asynchronous circuits has some problems. One problem is that asynchronous circuits tend to need more hardware for design implementation than their synchronous counterparts. This is especially true when completion detection techniques [6,7,8] are used to signal data validity to the recipient. The major problem for most design engineers may be their unfamiliarity with asynchro-

nous design methodologies. The methodologies are versatile [4] and not easy to cope with compared with the synchronous design methodology. This problem may become less critical when the methodology is simplified and is widely taught.

1.2 Problem Statement

Although people claim several potential advantages to using asynchronous circuits, as stated in the previous section, and some advantages are obvious, others need further formal verification. For example, since asynchronous circuits feature no global clock, clock–related problems occurring in synchronous circuits can be avoided naturally. That is, the clock skew, and clock–caused noise, power consumption and heat generation will not happen in asynchronous circuits. Moreover, the properties of modularity and composibility are obvious for certain classes of asynchronous design methodologies, such as micropipelines [9], trace theory [10,11] and communicating processes compilation technique [12]. As to the issue of the advantage of lower power consumption for asynchronous circuits, it is theoretically true since only the active portion consumes power. Some models have been used to successfully estimate power consumption in asynchronous circuits [13,14]. However, neither complete analysis nor extensive experiments have been performed showing that asynchronous circuits consume less power when hardware overhead is taken into account. For example, hardware is almost double if the dual–rail encoding method of completion detection technique is employed to indicate data validity [6,7]. Regarding performance, it is commonly believed that the performance (average throughput) of an asynchronous pipeline implemented with variable stage delay is better than that of its counterpart implemented using synchronous methodology. No work is done to formally prove that the above statement is always true under all circumstances, and is applicable to a general circuit.

In general, there are two metrics of measuring the performance of an asynchronous pipeline. One is throughput bounds (upper and lower) and the other is average

throughput. *Throughput bounds* limit a circuit's worst and best throughput allowed in an application, while *average throughput* defines the average performance. As will be seen later in this thesis, the performance (average throughput) of asynchronous pipelines depends on stage–delay patterns which might be totally random. Since performance investigation from the probability point of view is beyond the scope of this thesis, we will not propose any design guidelines for implementing circuits with optimized average throughput; instead, we will discuss some interesting circuit behavior which results from having different stage–delay patterns. Note that since the method adopted in this thesis is a deterministic approach, average throughput discussed here is limited to a finite input data set. That is, the conclusion regarding average throughput drawn in this thesis can not be generalized to the case of an infinite input data set. In other words, it is not the intention of this thesis to prove or disprove that the performance (average throughput with an infinite input data set) of an asynchronous pipeline implemented with variable stage delay is better than that of its counterpart implemented using synchronous methodology. Finding the throughput bounds is more interesting to us and will be addressed in more detail in this thesis. Also, it is well–known that the performance of synchronous pipelines is limited by the maximum physical (propagation) delay of combinational logic in each stage. Then, what determines the performance of asynchronous pipelines? In this thesis, we are particularly interested in solving some of the problems regarding the performance issues of asynchronous systems. Without considering whether the potential advantages of asynchronous circuits are true or not, many research and development projects on asynchronous microprocessors have been completed or are being undertaken [15,16,17,18]. These works also familiarize designers with asynchronous design methodologies.

1.3 Thesis Contribution (Goal) and Organization

Better performance has been one of the main motivations behind the recent resurgence of interest in asynchronous circuits (no matter whether it is always true or not). We are particularly interested in the performance of pipelines since they are used extensively in digital systems. Why is performance information for asynchronous circuits so important? It can be described from two different points of view. In the analytical phase, performance analysis can generally verify if a circuit meets specified timing constraints, such as time duration between consecutive output data. This information plays an important role in determining the size of asynchronous First In First Out buffers (FIFOs) used as an interface, and the performance (and hence the probability of synchronization failure) of communication bandwidth between asynchronous and synchronous systems, or between synchronous (clocked) systems driven by different clocks (or the same clock but with clock skew) [19,20]. In the design phase, with timing analysis results, we are able to derive or observe some design rules (guidelines). These rules help the design satisfy performance specifications. Instead of the trial–and–error approach, these guidelines may also help designers reach the optimized performance in shorter design cycle time.

An algorithm developed for finding the exact upper and lower bounds on the separation time of events in a certain class of process graph [21,22] might be a good candidate for the application of Computer–Aided Design (CAD) tool development and circuit synthesis. However, some transformations and complex mathematical analyses, like graph decomposition for infinite unfolded process graphs, must be employed to reach exact bounds. These transformations and mathematical analyses tend to block designers from visualizing what factors really affect the performance of asynchronous circuits. This is especially true for most practicing engineers who are familiar with synchronous designs and lack experience in asynchronous design. We believe that one of the reasons why most engineers are reluctant to engage in asynchronous designs is

that it is difficult to analyze asynchronous circuits (in terms of performance), not to mention the difficulty in designing a high performance asynchronous circuit. By first investigating the performance of a simple linear micropipeline, we hope to help engineers to understand the performance of asynchronous circuits and to acquire insight into what influences their performance.

In summary, the goals of this thesis are:

(1) To provide a simple method for finding the throughput bounds (approximation) of a general micropipeline, given stage–delay bounds;

(2) To acquire average throughput rate, given stage–delay patterns (sequences);

(3) To compare performance–related issues of asynchronous designs with those of synchronous designs;

(4) To provide a design guideline and several approaches to meet performance bounds specifications.

The approach to meeting the goals of this thesis is stated as follows. This thesis is organized into the following chapters. The motivation, problem and goal are discussed in this introductory chapter. A reviewing of previous works regarding the performance of asynchronous pipelines is done in Chapter 2. Chapter 3 gives the introductory materials about micropipelines and the delay modeling of C–elements. Based upon this C–element delay modeling, Chapter 4 investigates the performance of fixed stage–delay linear micropipelines. Although performance results for this type of pipeline have been reported elsewhere [23,24,25], it is still covered in this chapter to demonstrate that our approach can lead to the same results. With slight modification, some theorems developed for fixed stage–delay micropipelines are applied to micropipelines with variable stage–delays in Chapter 5. Using the knowledge and results from previous chapters, some interesting phenomena are depicted and compared with synchronous designs through several examples. Chapter 6 summarizes this comparison. Immediately following is Chapter 7, describing the performance calculation for two–dimensional micropipe-

lines based on the concept and results shown in Chapter 5. Chapter 8 provides several examples to compare exact throughput bounds with approximate bounds obtained using our approach. Finally, we conclude our approach in the last chapter.

1.4 Summary

Asynchronous circuits are claimed to have some advantages over synchronous circuits. The issue of better performance is one of these potential advantages. This thesis tries to explore some performance–related problems for asynchronous circuits. The focus is trying to avoid too complex mathematics and algorithms that may possibly block the designer's insight in finding the performance–influencing factors, and to provide a simple method for obtaining approximate performance. This thesis is capable of:

(1) providing a simple method for finding the throughput bounds (approximation) of a general micropipeline, given stage–delay bounds,

(2) acquiring average throughput rates, given stage–delay patterns (sequences),

(3) comparing the performance–related issues of asynchronous designs with those of synchronous designs,

(4) providing a design guideline and several approaches to meet performance bounds specifications.

2. LITERATURE REVIEW

As mentioned in the previous chapter, the focus of this thesis is on issues related to the performance of asynchronous pipelines. This chapter presents a review of related literature. The main purpose of this review is to demonstrate the current research status of this particular performance topic by the other researchers. Only their models are introduced. Their algorithms for obtaining performance results will not be discussed. Interested readers may go to references cited in this thesis for details. The model and approach presented in this thesis are believed to be much easier than the other works with the sacrifice of accuracy.

2.1 The Operation Of A C–element

As will be mentioned in the next chapter, Sutherland [9] has proposed many basic logic modules for the design of event–control mechanisms in micropipelines. The C–element is one of these building modules. In reality, a C–element [26] is frequently used in asynchronous designs. Figure 2.1shows two different symbols for a two–input C–element. The upper left symbol is often used at lower level representations while the upper right symbol is used at higher level representations. Use of the lower level representation implies an interest in absolute logic level at terminals. On the other hand, higher level representations are used when the distinction of absolute logic level at terminals is not required or necessary. The operation of a C–element in terms of absolute logic level at terminals is demonstrated as follows. The output signal of a C–element is high only when both input signals are high; the output signal becomes low only when both input signals become low. Otherwise, the output signal of a C–element remains in its previous state. The waveforms in Figure 2.1 summarize the operation of a C–element as stated above. Although *In1* becomes High at point ***A***, *Out* will not change until *In2* also changes to High (point ***B*** in the figure). The time that *In1* waits for *In2* to change is

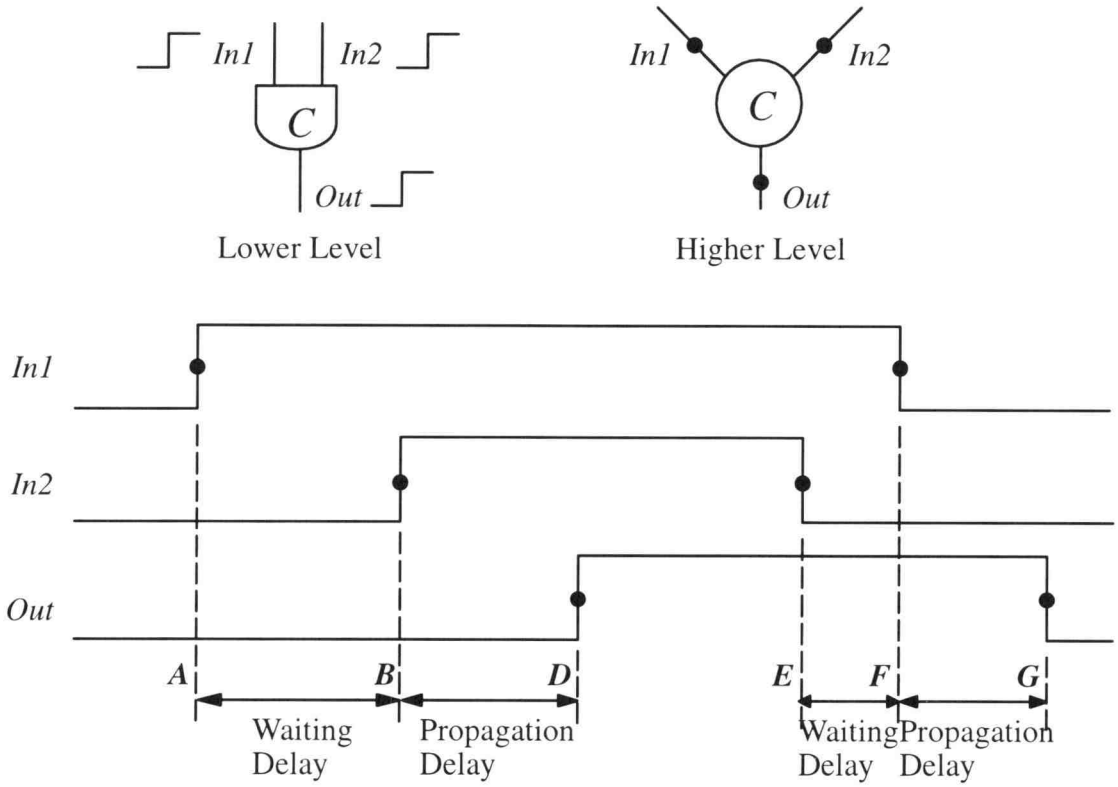


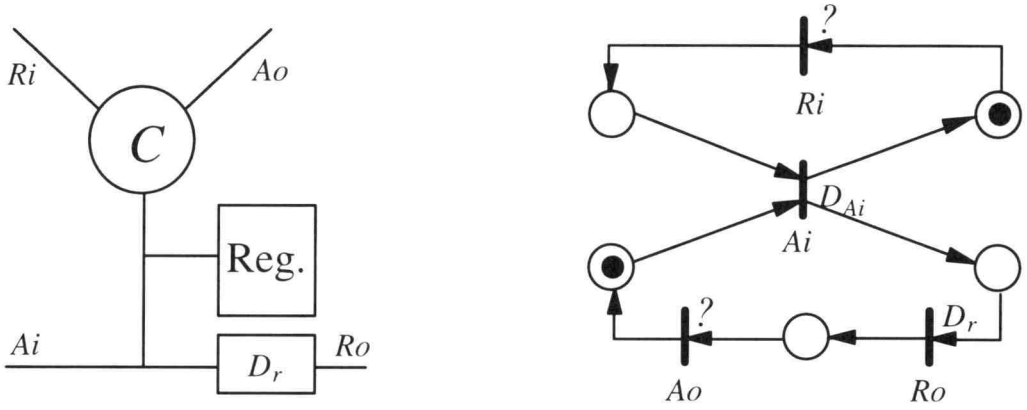
Figure 2.1: The operation of a C–element.

called a *waiting delay*. In reality, the logic level at *Out* terminal will not immediately respond to both input terminal changes at point ***B***. Instead, *Out* will not change until point ***D***. This is obvious since the signal needs time to propagate through C–element. This time is called a *propagation delay*. Similar arguments can be applied to the waveform changes at point ***E***, ***F*** and ***G***. In summary, waiting and propagation delays can fully describe the waveform relationship between input and output terminals of a C–element. In general, the waiting delay is not zero, the case in which both *In1* and *In2* change at the same time.

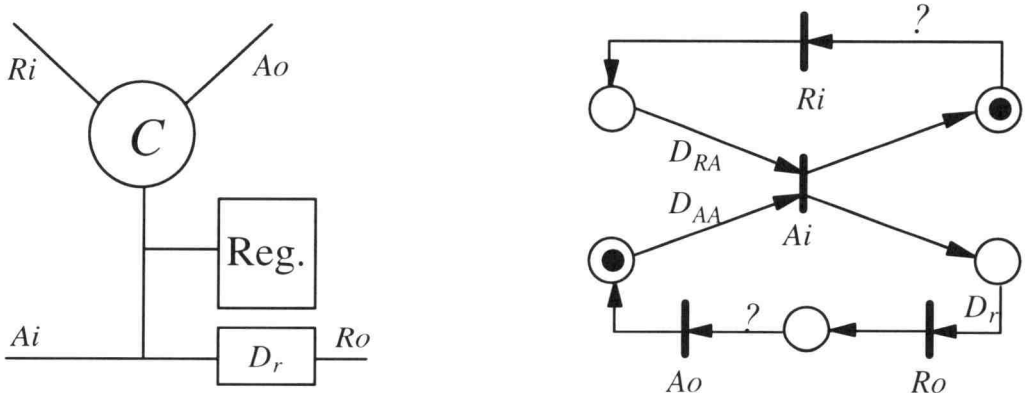
The C–element at higher level representation utilizes tokens instead of absolute logic value to indicate its operation. The solid dots in Figure 2.1 represent the tokens. That is, the higher level token symbolizes both low–to–high and high–to–low signal transitions. With token representation, the firing rule for C–element is stated as follows. A C–element cannot fire until both input tokens arrive. Once it fires, both input tokens are consumed and an output token is generated. Therefore, the waiting delay is the time that one input token must wait for the other input token to arrive. Once both input tokens arrive, the time from now until the output token is generated is called the propagation delay. A more formal modeling of C–elements will be discussed in the next chapter.

2.2 Timed Petri Net Modeling

Most performance analyses of asynchronous circuits use a timed Petri net [25] approach as a base to develop into different forms. Figure 2.2 shows circuits and their corresponding Petri net modeling. A Petri net contains two types of nodes — place and transition. *Places* in a Petri net are drawn in circles and represent conditions; *transitions* are drawn in bars and represent events. A token, symbolized by a solid dot, at a place indicates the holding of the condition (state) of the place. The firing rules of Petri nets are summarized as follows.



(a)



(b)

Figure 2.2: Circuits and their corresponding timed Petri net model.
(a) Approach by Ramamoorthy and Ho.
(b) Approach by Wuu and Vrudhula.

(1) A transition is enabled if and only if each of its input places has a token;

(2) A transition can fire only if it is enabled;

(3) Once a transition fires, a token is consumed from each of its input places and a token is generated at each of its output places.

We assume that each place is allowed to occupy at most one token. Therefore, a transition will be enabled but not be able to fire if each of its input places has a token and one of its output places also has a token.

2.2.1 Approach By Ramamoorthy And Ho

The Petri net model extended to include the notion of time is called the *timed Petri net*. Ramamoorthy and Ho [25], attach the execution time to each transition. Accordingly, if an execution time for a transition is t, this implies that when a transition initiates its execution, it takes t units of time to complete it. This kind of time modeling seems ambiguous since, from the circuit point of view, the event (signal) transits instantaneously and does not take some units of time for execution. Only a processing element needs time for execution (generally speaking, this refers to propagation delay). Most likely, the execution time of a transition, as defined in this approach, implies that the propagation delay of a processing element which has an output terminal carries an event (signal) corresponding to that transition. For example, for the transition A_i in Figure 2.2(a), its execution time D_{Ai} refers to the propagation delay of the C–element. No waiting delay is included in D_{Ai}. The question marks "*?*" in the same figure mean unknown execution time which depends on the other circuits it connects with. Once the timed Petri net modeling the corresponding circuit is constructed, some theorems developed in this approach are applied to find the maximum performance (minimum cycle time). Also note that this approach only allows constant transition execution time.

2.2.2 Approach By Wuu And Vrudhula

Basically, the approach by Wuu and Vrudhula [24] follows the work by Ramamoorthy and Ho [25]. Their way of associating the time attributed to Petri net, however, is different. Wuu and Vrudhula's work is much closer to circuit operation. They define the delay from a place (state) to a transition (event) and label this delay on the arc linking the corresponding place and transition. This delay time represents the minimum time interval from when the condition is satisfied to when the transition is activated. The minimum time in their definition refers to the propagation delay. For example, in Figure 2.2(b), $D_{RA} = D_{AA}$ and equals the propagation delay of the C–element since the minimum time would be the time that the waiting delay is zero.

2.3 Dependency Graph Modeling

The work proposed by Williams [27] uses a directed graph, which is a simplification of the more general timed Petri net, to model a circuit and, further, to find its performance. The nodes of the directed graph correspond to specific rising or falling transitions of circuit components, and the arcs (edges) depict the dependencies of each transition on the output of other components. The resulting directed graph is called a *Dependency Graph* (DG). The delay of each transition in a DG is represented by a value attached to the corresponding node. Figure 2.3 is an example of circuit modeling using DG. Note that a bubble (or circle) attached to the input or output terminal of a C–element represents that the logic level of the corresponding terminal is inverted. *Folded Dependency Graph* (FDG) can be used to model a special ring architecture which uses the same function–blocks and circuit configuration for all stages. FDG modeling is possible because the nodes can represent the same transition in all stages and each arc is annotated with an integer weight giving the offset in stage indices to which the transition dependency refers. The work by Williams focuses on the performance of four–phase (since DG/

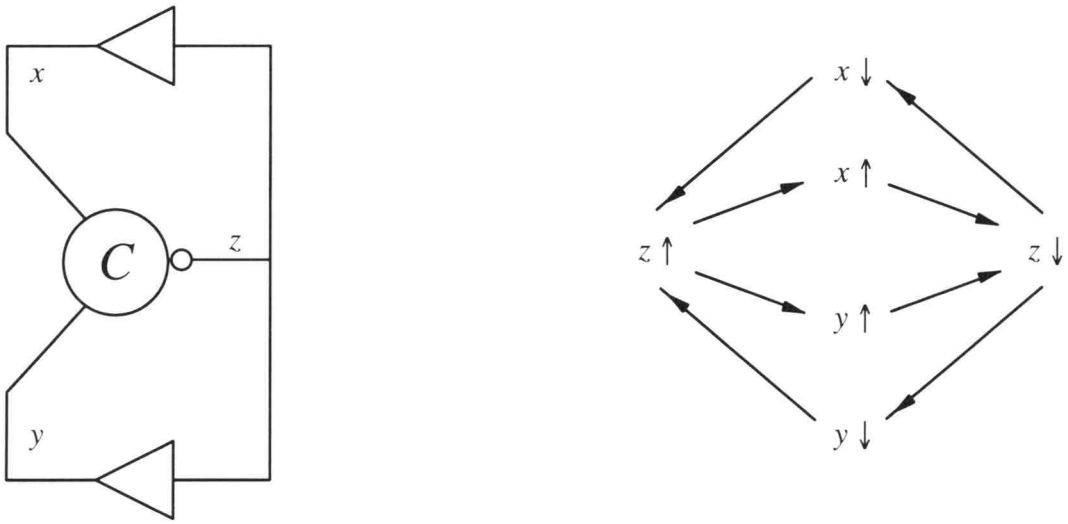


Figure 2.3: A circuit and its corresponding Dependency Graph.

FDG defines rising or falling transitions on nodes) ring architecture. His contribution is in obtaining the relationship between the performance and the number of tokens in a ring. The timing information attached to each node in DG/FDG is constant.

2.4 Event Rule Modeling

Another approach to circuit modeling is the *Event–Rule* (ER) model (system) which was introduced by Burns [23]. An ER system is defined as follows.

E: a set of events (the event specified here actually refers to the transition), and

R: a set of rules defining timed constraints between the events defined in E.

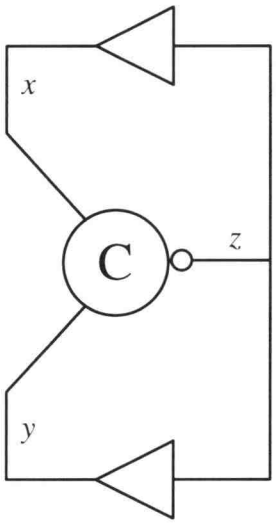
Each element $r \in R$ is written $c \overset{\alpha}{\mapsto} d$,

where $c \in E$ is the source of r,

$d \in E$ is the target of r, and

$\alpha \in [0, +\infty)$ is the delay of r.

Figure 2.4 shows a circuit and its corresponding ER system. The delay definition is not clear. For example, the two delays within the dotted box in Figure 2.4 have the same amount $\tau_{Z\downarrow}$. In general, this would be true only when the delay defined in an ER system is the minimum delay that causes a target event to occur once a source event happens. Otherwise, these two delays would not be the same (e.g., if the waiting delay is taken into account). Based upon linear programming, the performance can be obtained for the circuit using ER modeling. Although Burns [23] proposed an approach for variable stage–delays by unfolding graphs, no explicit algorithm was mentioned on how to find the performance of infinite unfolded graphs. Recent publications by Hulgaard et al [21,22] have extended previous results to include the performance of infinite unfolded graphs. Since the exact upper and lower bounds on the separation time of events in a certain class of process graph can be found using their approach [21,22], their algorithm might be a good candidate for the application of CAD tool development and circuit



$$E = \langle x\uparrow, y\uparrow, z\uparrow, x\downarrow, y\downarrow, z\downarrow \rangle$$

$$R = \langle \langle x\downarrow, i-1 \rangle \overset{\tau_{z\uparrow}}{\mapsto} \langle z\uparrow, i \rangle,$$

$$\langle y\downarrow, i-1 \rangle \overset{\tau_{z\uparrow}}{\mapsto} \langle z\uparrow, i \rangle,$$

$$\langle z\uparrow, i \rangle \overset{\tau_{x\uparrow}}{\mapsto} \langle x\uparrow, i \rangle,$$

$$\langle z\uparrow, i \rangle \overset{\tau_{y\uparrow}}{\mapsto} \langle y\uparrow, i \rangle,$$

$$\langle x\uparrow, i \rangle \overset{\tau_{z\downarrow}}{\mapsto} \langle z\downarrow, i \rangle,$$

$$\langle y\uparrow, i \rangle \overset{\tau_{z\downarrow}}{\mapsto} \langle z\downarrow, i \rangle,$$

$$\langle z\downarrow, i \rangle \overset{\tau_{x\downarrow}}{\mapsto} \langle x\downarrow, i \rangle,$$

$$\langle z\downarrow, i \rangle \overset{\tau_{y\downarrow}}{\mapsto} \langle y\downarrow, i \rangle \rangle$$

Figure 2.4: A circuit and its corresponding Event–Rule system.

synthesis. However, some transformations and complex mathematical analyses, such as graph decomposition for infinite unfolded process graphing, must be employed to reach exact bounds. These transformations and mathematical analyses tend to block designers from visualizing what factors really affect the performance of asynchronous circuits.

2.5 Other Works

One work which explored asynchronous circuits with both fixed and random delays was proposed by Greenstreet [19]. This work is primarily a discussion of the throughput and utilization of two–phase, circulating pipelines (ring architecture). Thiele [28] developed an approach that can be applied to obtain the performance of a more general self–timed architecture. However, this approach also involves complex mathematics.

2.6 Summary

Current literature on the performance analysis of asynchronous circuits has been reviewed in this chapter. Some of the authors discussed their research on performance for special architecture. For example, two discussed pipelines with ring architecture: Williams for a four–phase signaling scheme [27] and Greenstreet for a two–phase [19]. Wuu and Vrudhula discussed linear architecture [24]. Others described methods which are applicable to more general architecture [22,25,28]. Ramamoorthy and Ho only allow constant delays in a circuit [25] and Hulgaard et al [22] and Thiele [28] involve too much complex mathematics, although these last two may be good candidates for the application of CAD tool development and circuit synthesis. The goal for this thesis is to propose an easier model and approach that can handle the performance (throughput bounds) of the general architecture of asynchronous pipelines (micropipelines) with variable stage–delays.

3. MICROPIPELINES AND C–ELEMENT MODELING

The micropipeline is one of the many asynchronous design methodologies [4]. In this thesis, we are particularly interested in the micropipeline because its modularity and composibility feature facilitates hierarchical design, which is preferred in the design of complex systems by a team. Besides, it is simple and easy to understand. The C–element is a key module in a micropipeline, coordinating the operation (synchronization) of adjacent stages. In this chapter a brief review of the design and operation of a micropipeline is presented. An introduction to C–element modeling is included in this chapter, laying the ground for the performance evaluation of micropipelines discussed in following chapters.

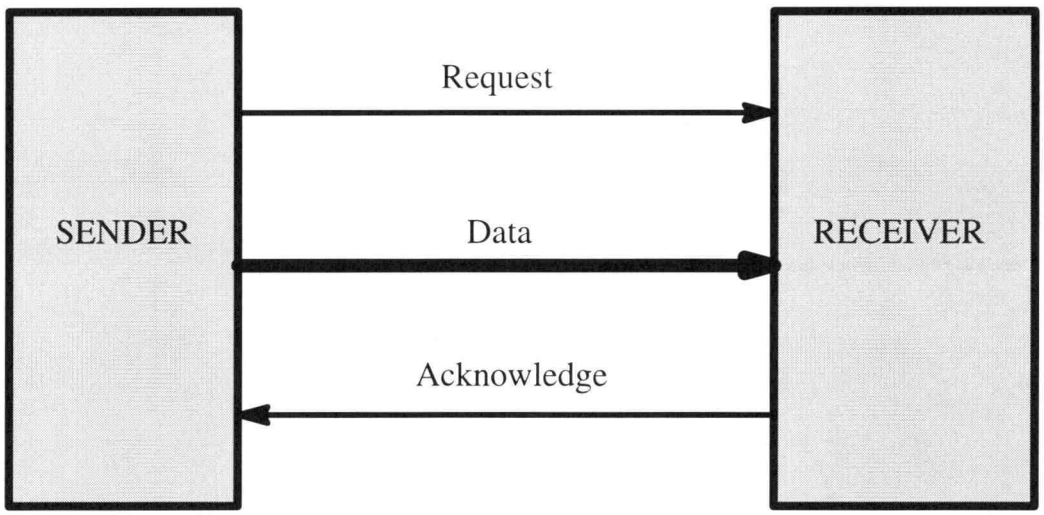
3.1 The Two–Phase Bundled Data Convention

Just as the signals in synchronous systems, the signals in micropipelines have two types — data signals and control signals. The way of designing a data path, the place through which data signals flow, is the same for both synchronous systems and micropipelines. The way of dealing with control signals, however, is totally different between these two design approaches. Therefore, in the following text we will focus only on a discussion of control signals. In a conventional synchronous design philosophy, absolute logic level (high or low) is assigned to each signal and different logic values have different meaning for each signal. In micropipelines, signal transition is of more interested than absolute logic level. Moreover, both low–to–high and high–to–low signal transitions have the same meaning to circuit operation. That is, both rising and falling edges of a signal may trigger circuit operations. Since the distinction of low–to–high and high–to–low signal transitions is not necessary, a more general term *event* is used to replace the signal transition.

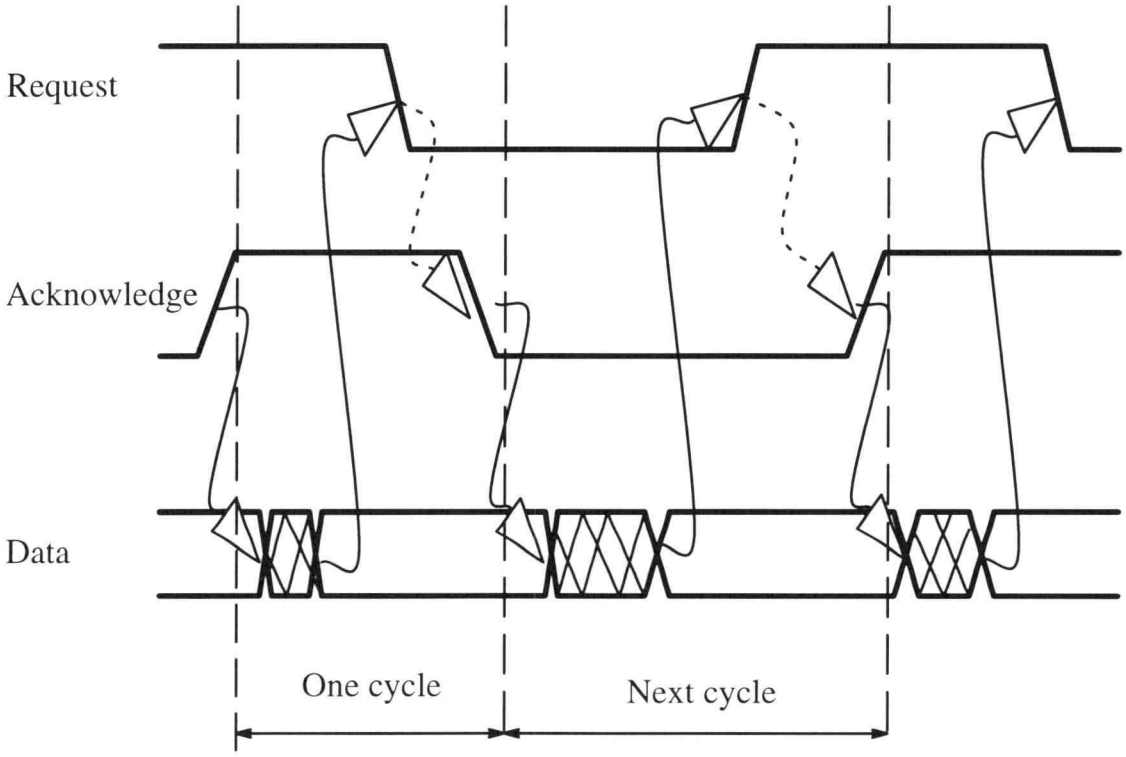
All of the module communications within a micropipeline follow the same basic connection, as demonstrated in Figure 3.1(a). n a sender and receiver environment, a handshaking protocol consisting of two control wires and many data wires is used. One of the control wires is called request and the other one acknowledge. The signals on both request and acknowledge wires must be designed to follow the transition–signaling rule. For the signals on data wires, absolute logic meanings are still preserved such that the conventional combinational logic designs can still be used in micropipelines. To ensure the correct circuit operation in Figure 3.1(a), the *two–phase bundled data convention* must be strictly observed. This convention requires that a complete operation (called *cycle*) between adjacent stages have two phases: the sender phase sends a request signal to start the operation and receiver phase sends an acknowledge signal to close the operation. "Bundled data" indicates that data and request wires must be treated as a bundle: data must be stable before the request signal arrives at the receiver end. The complete handshaking between sender and receiver in a cycle is described as follows. Sender generates valid data on the data wires and then issues a request event to signal receiver that the data are available. The receiver takes the data whenever it is ready and then produces an acknowledge event to tell sender that the data have been received. At this point, sender can put new data on data wires and the whole procedure repeats. Note that data can not be changed before acknowledge event is issued. The relative signal timing diagram is illustrated in Figure 3.1(b).

3.2 Linear Micropipelines

Due to the modularity and composibility feature, the basic control circuit for micropipelines can be constructed easily and readily by stringing each stage together directly. The resulting pipeline is called a *linear micropipeline*. From the higher–level sender/receiver point of view, each stage plays both roles (sender and receiver) in micropipelines, as shown in Figure 3.2(a). For example, stage two performs the receiver role

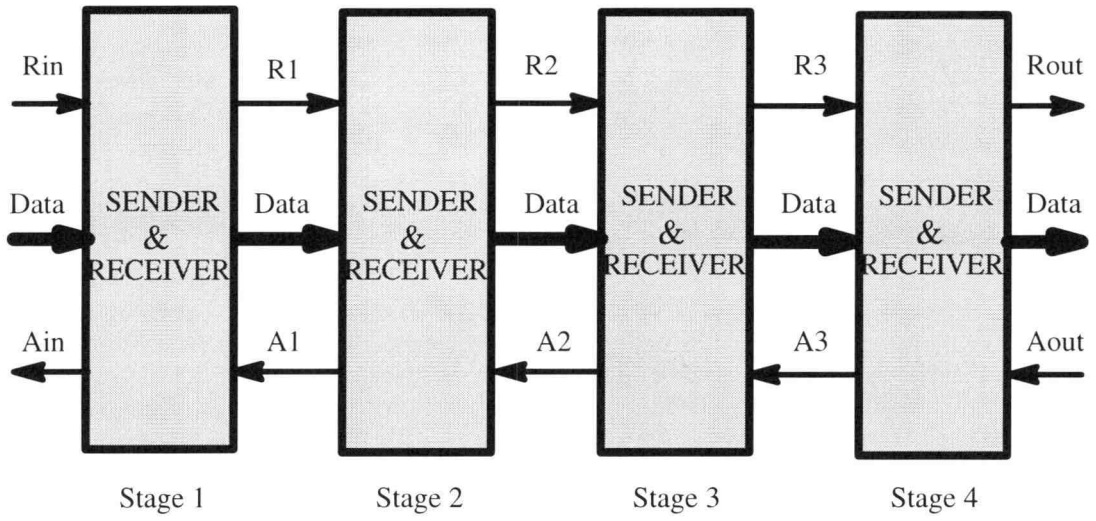


(a)

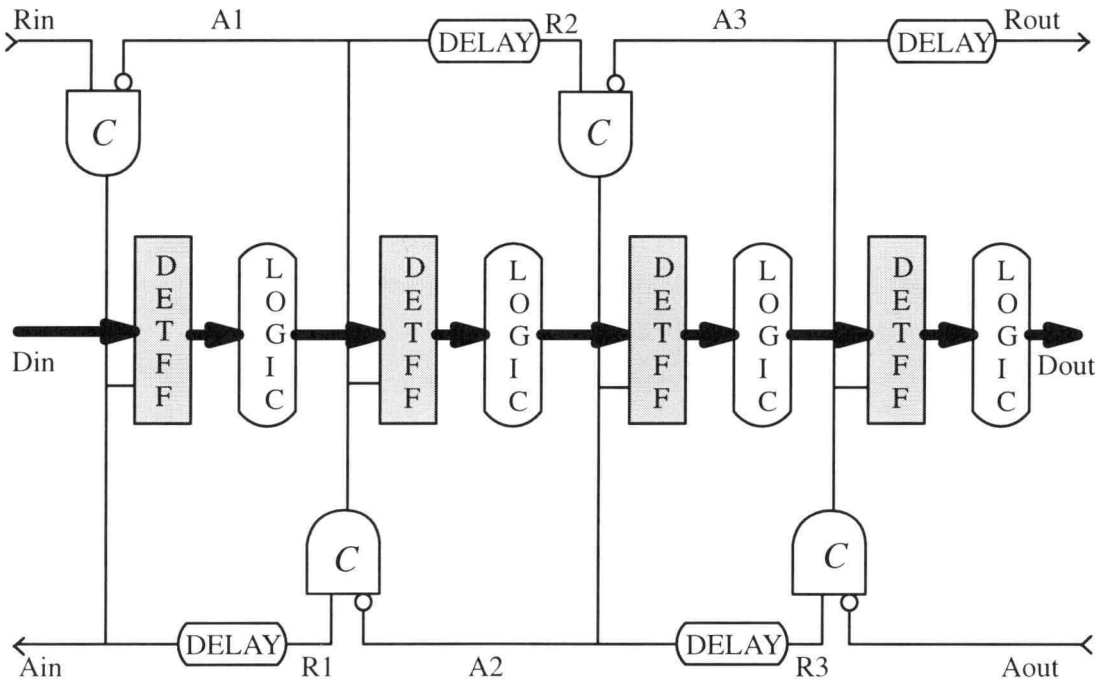


(b)

Figure 3.1: Two–phase bundled data convention.
(a) Module interface in micropipelines.
(b) Timing diagram.



(a)



(b)

Figure 3.2: A linear micropipeline.
(a) Abstract level.
(b) Circuit level.

relative to stage one when stage one tries to send data to stage two; after stage two obtains data from stage one, it will pass data (which may be modified) to stage three as a sender. All data and control signals transferred between consecutive stages must follow the two–phase bundled data convention.

The mechanism for fulfilling the two–phase bundled data convention requirement in a micropipeline is very straightforward. It requires only strings of bubbled C–elements (a C–element with a bubble on one of its input terminal, indicating that the signal is inverted) with appropriate interconnection, as illustrated in Figure 3.2(b). Observe that only odd number of bubbles (one in the basic circuit) are allowed in every loop around which events flow. The purpose of the bubble in each loop is to form oscillation such that data fed into the input end can be bubbled through to the output end automatically without other control circuits. Although the absolute state of a control signal does not matter, its state relative to the other related signals (e.g., two input terminals for a bubbled C–element) does matter. Therefore, the state (output) of current bubbled C–element can be described in terms of the states of predecessor and successor bubbled C–elements:

IF the states of predecessor and successor are different

THEN copy predecessor's state

ELSE remain in the present state

The control circuit is stable only when one of the following three conditions is satisfied:

(1) All of the bubbled Muller C–elements are in the same states. This corresponds to an empty pipeline. New data can be fed into this circuit.

(2) Alternate stages are in opposite states. This corresponds to a full pipeline. No more data is allowed to enter this pipeline until some data have been consumed (removed).

(3) Stages near the input end have the same state, and stages near the output end have alternate states. This corresponds to a partly filled pipeline. New data can only fill empty stages.

The stage states (output states of bubbled C–elements) that are not in one of these state conditions cause the circuit to be unstable. The circuit will not stop (idle) in an unstable state; it will continue to propagate through automatically until one of the above conditions is reached. Once a stable condition is reached, the circuit becomes idle (theoretically, consuming no power) until a new datum enters or a datum moves out to destroy the stability.

Each delay element in Figure 3.2(b) ensures that the request signal and data are bundled in a way that valid data will always arrive at the next stage prior to the arrival of the corresponding request signal. Therefore, the value of each delay element must be equal to or greater than the worst–case delay of the combinational logic circuit within that stage. The original data storage element used in micropipelines is a two–control–wire (*capture* and *pass*) register. The capture signal latches present data and the pass signal allows new data to enter the storage element. In Figure 3.2(b), a Double–Edge–Triggered Flip–Flops (DETFF) [29] is employed instead because it is single wire (easier to understand than two wires operation) and works as a regular D Flip–Flop except that it is triggered at both rising and trailing edges of a control signal. If all internal processing logics are removed, the whole circuit shown in Figure 3.2(b) works like a FIFO.

3.3 Delay Modeling of C–elements

As stated in the previous section, a C–element coordinates two adjacent stages in a micropipeline, such that handshaking between these two stages is performed appropriately. Since a C–element is the key module in a micropipeline, we will restate its operation and depict it in Figure 3.3 (a slight modification of Figure 2.1). The output signal of a C–element is high only when both input signals are high; the output signal

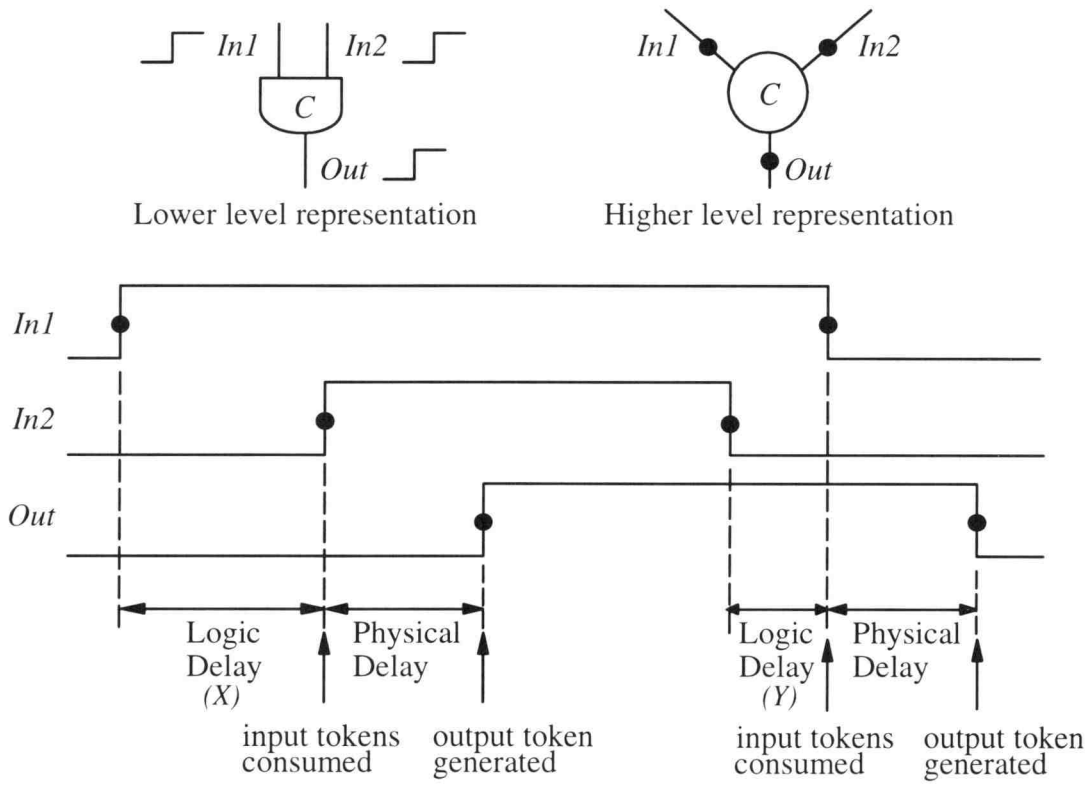
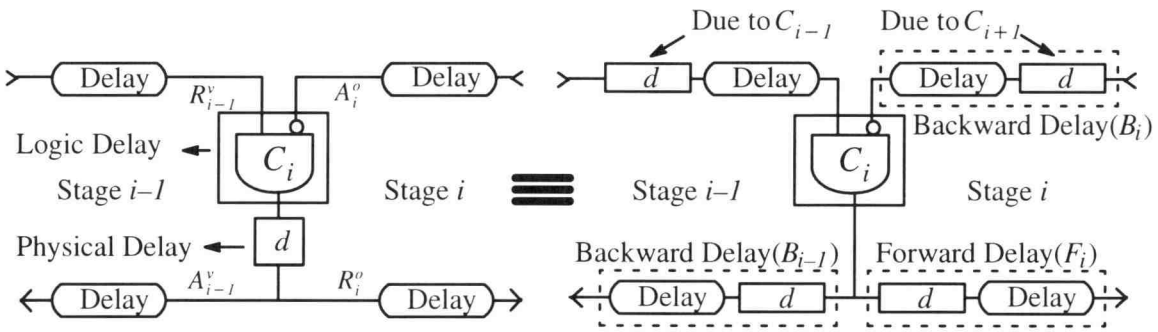


Figure 3.3: The operation of a C–element (modified).
A higher–level token (solid black circle) on each arch represents a lower–level transition for each signal.

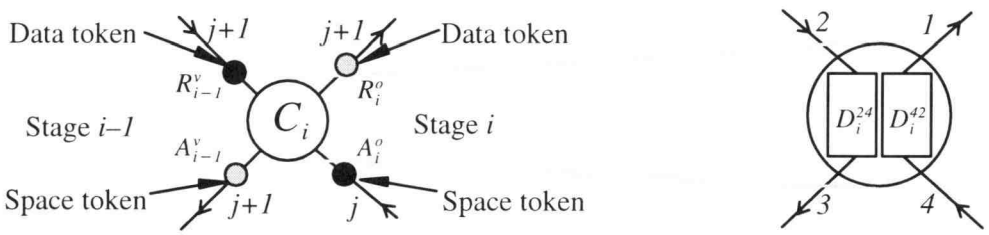
becomes low only when both input signals become low. Otherwise, the output signal of a C–element remains in its previous state. Since micropipelines use two–phase transition signaling methodology (i.e. low–to–high and high–to–low transitions have the same meaning to circuits) and one of the C–element inputs used in micropipelines is inverted, an absolute signal level representation (high or low) is avoided to prevent confusion. This is especially true when we are only interested in the performance issue. As a result, a token is introduced to depict a signal transition (both low–to–high and high–to–low). Both token and event refer to the signal transition although token is most widely used when the performance issue is discussed. With token representation, the C–element firing rule can be restated as follows. A C–element cannot be enabled (fired) until both input tokens arrive. Once it is enabled, input tokens are consumed and an output token is generated.

In order to model the flowing of tokens in a micropipeline, the physical (or propagation) delay and logic (or waiting) delay shown in Figure 3.3 should be reflected in the C–element model given in Figure 3.4. *Physical delay* in Figure 3.3 is defined as the duration from the time input tokens are consumed to the time output tokens are generated. Physical delay in our C–element model is denoted as d in Figure 3.4(a). This delay can be further merged with the (combinational logic) stage–delay of the micropipeline in both the directions of the token path; the result is named *forward delay* (F_i) and *backward delay* (B_i), as shown at the right hand side of Figure 3.4(a). The *total stage–delay* D_i for stage i in a micropipeline is defined as $D_i = F_i + B_i$.

Logic delay in Figure 3.3 is defined as the time duration that one input token must wait for the other input token in order for the C–element to fire. Figure 3.4(b) shows our C–element logic delay model. The arcs for the C–element in Figure 3.4(b) correspond to the arcs for the C–element in Figure 3.4(a) with the same arc labeling R_i^o, R_{i-1}^v, A_{i-1}^v and A_i^o. In order to simplify the logic delay representation, we will map these arcs to *1*, *2*, *3*, and *4*, respectively, as illustrated in the graph at the right hand side of



(a)



(b)

Figure 3.4: Modeling of physical and logic delays of a C–element.
(a) Physical delay modeling.
(b) Logic delay modeling.

Figure 3.4(b). The token on arc R^{v}_{i-1} is called a *data token* since it is the token issued from the previous stage to flag the valid data. On the other hand, the token on arc A^{o}_{i} is named a *space token* because it is generated from the next stage to acknowledge space availability (indicating the previous datum has been received by the next stage). Note that the C–element in Figure 3.4(b) has four terminals but in Figure 3.3 it has only three terminals. In terms of functionality, these two C–element representations are the same because, physically, the terminal R^{o}_{i} and A^{v}_{i-1} share the same wire. The C–element representation depicted at Figure 3.4(b) is adopted for our performance derivation in the following chapters.

In Figure 3.4(b), logic delay D^{24}_{i} represents the amount of time that a data token waits for a space token to appear and logic delay D^{42}_{i} is the time period that a space token waits for a data token to arrive. It is important to know that logic delay is a function of time (tokens) in general. To maintain notation consistency throughout a whole pipeline, we define that the *(j+1)th* data token (dark color) is always matched with the *jth* space token (dark color). After a C–element fires, the two output tokens (gray color) are numbered as *j+1s*. The output token produced at the upper right arc is called a data token; the output token generated at the lower left arc is named a space token. That is, a data token always travels forward at the upper circuit path to indicate that intermediate data (result) is moving toward the output end to produce a new output result and a space token always travels backward at the lower circuit path to indicate that an empty space (stage) is moving toward the input end to accept new input data. Although different tokens at different locations (arcs) have different names, there is no difference among these tokens in terms of circuit operations. In general, logic delay should be represented as $D^{24}_{i}(index1, index2)$ and $D^{42}_{i}(index1, index2)$, where *index1* refers to the appearing sequence of its input data token and *index2* refers to the appearing sequence of its input space token. According to the logic delay definition and firing rule for C–element, $D^{24}_{i}(j+1, j) * D^{42}_{i}(j+1, j) = 0$ for all of the j. For example, let *In1* and *In2* in

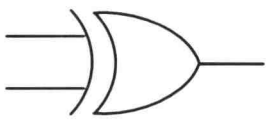
Figure 3.3 represent the data token and space token of Figure 3.4(b), respectively. Since the first data token (rising edge) arrives earlier than the first space token by X (logic delay), then $D_i^{42}(j+1,j) = 0$ and $D_i^{24}(j+1,j)$ equal to X. Since the data token representing the trailing edge of *In1* arrives later than the space token representing the trailing edge of *In2* by Y (logic delay), then $D_i^{24}(j+1,j) = 0$ and $D_i^{42}(j+1,j)$ equal to Y. If both tokens arrive simultaneously, then $D_i^{24}(j+1,j) = D_i^{42}(j+1,j) = 0$. For more compact representation, the second token index of a logic delay representation is omitted in this thesis. The micropipelines under investigation are assumed initially empty; therefore, $D_i^{24}(1,0) = D_i^{24}(1) = 0$ and $D_i^{42}(1,0) = D_i^{42}(1)$ is undefined for all of the i. Note that all the delays, including logic delays, mentioned in this thesis are greater than or equal to zero.

3.4 Event Logic Modules

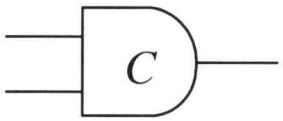
We have introduced basic control circuits, linear micropipelines, previously. For diverse applications of micropipelines, more event logic modules are required as basic elements from which to construct more complicated designs of event logic circuits. Event OR, event AND, Toggle, Select, Call and Arbiter, as shown in Figure 3.5, are the basic modules discussed by Sutherland [9].

(1) ***Event OR:*** If any input of this module changes its state, the corresponding output also changes its state. In terms of events, the arrival of any input event generates an output event. This works exactly the same as a conventional XOR gate so its symbol is adopted as event OR module.

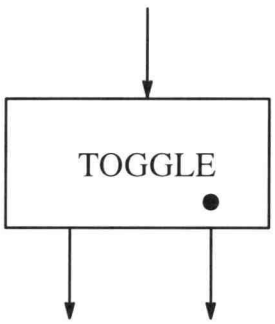
(2) ***Event AND***: Event AND module is the C–element which has been introduced previously.



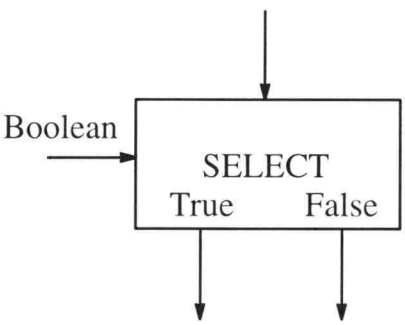
Event OR (XOR)



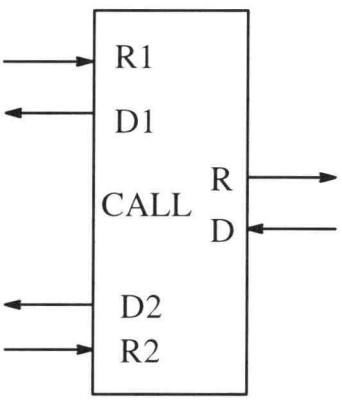
Event AND (C–element)



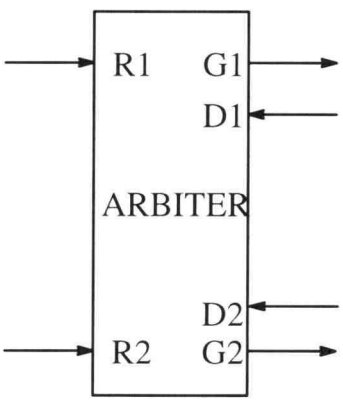
Toggle



Select



Call



Arbiter

Figure 3.5: Depiction of some event logic modules.

(3) ***Toggle:*** Toggle generates an output event alternately, starting with the dotted terminal. It responses to events at its input after the initial master clear signal which is not shown in the figure.

(4) ***Select***: Select steers an input event to the corresponding output according to the Boolean value at the other input terminal. Note that the Boolean value must arrive before the input event.

(5) ***Call***: This hardware Call module serves the role of memorizing the subroutine return address. It remembers which input events, R1 or R2, have called the procedure (the event at R is generated). After the completion of procedure execution (the event at D is generated), it returns a matching done event on D1 or D2. The Call works properly only when the current call has been completed before the next call occurs.

(6) ***Arbiter***: An arbiter is used to guarantee mutually exclusive access to shared or protected resources. It grants requests (token on R1 or R2) by generating a token on the G1 or G2 terminal. The service is performed using the first–come first–served rule. When one of the input events grants the service, the other event (if any) will be delayed automatically by this module until the current one is done.

We will discuss the performance of micropipelines constructed by these modules after the performance of linear micropipelines is studied in the following chapters.

3.5 Summary

Micropipelines feature transition signaling schemes. That is, both the low–to–high and high–to–low control–signal transitions (events) have the same meaning to circuit operation. To ensure correct functioning of a circuit, micropipelines must follow the two–phase bundled data convention. Since micropipelines deal with events in designing the control path, some basic but important event logic modules are provided for convenience. Once each module is designed appropriately and the bundled data convention is strictly observed, direct interconnection is allowed between modules to form a

larger and more complex module or system. The delay modeling of a C–element is also included in this chapter for the performance calculation of linear micropipelines, which will be discussed in the following chapters. The performance for a more general micropipeline can be obtained based upon the performance approach for linear micropipelines.

4. THE PERFORMANCE OF A LINEAR MICROPIPELINE WITH FIXED STAGE–DELAY

In this chapter, we are interested in finding out how a linear micropipeline performs with fixed stage–delay. Although performance results for this type of pipeline have been reported by others [23,24,25], the topic is still covered in this thesis to demonstrate that our approach can lead to the same results. Besides, some properties, definitions and theorems developed in this chapter are applicable to the performance analysis of a more general micropipeline discussed in later chapters. The approach adopted is based mainly on the delay modeling of C–elements, as described in the previous chapter.

4.1 Definitions And Theorems

Since two logic delays are defined for each C–element and each logic delay belongs to a different "loop," the modeling of linear micropipelines can be derived straightforwardly. Based upon the C–element delay model, the performance analysis of micropipelines with fixed stage–delay is presented here. Fixed stage–delay implies that both the forward delay (F_i) and backward delay (B_i) for a stage i are independent of time (token), i.e., they are constant for all js, where j is a token sequence index. Hence, total stage–delay, D_i (= F_i + B_i), for a stage i is also a constant. Forward delay, F_i (excluding C–element physical delays d in [9]), is constant because the "delay" element of the *ith* stage in Figure 3.4(a) is chosen as the maximum combinational logic delay, such that the bundled data convention is satisfied. The backward delay (B_i) is included in our discussion for completeness. In our C–element model, B_i also contains physical delay d. Before we conclude with the results, some definitions are established first. From these definitions, we conclude with a few results as theorems and corollaries. The micropipeline we are interested in is a m–stage pipeline, as shown in Figure 4.1.

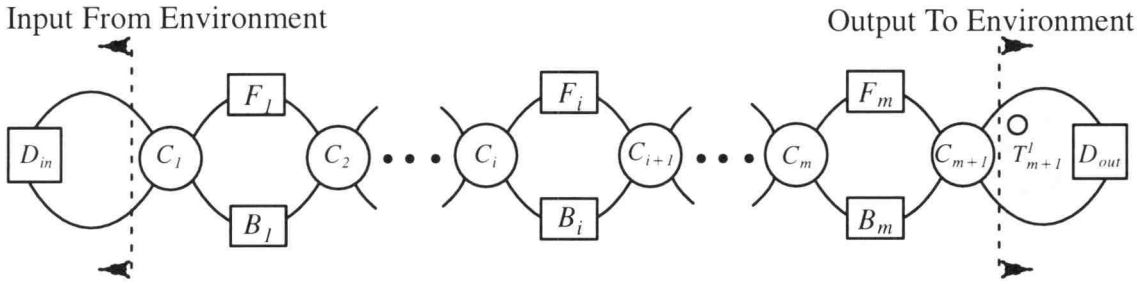


Figure 4.1: A m–stage linear micropipeline.

Definition 1: A *loop i* is defined as a circle around which a token will travel within a stage *i*. The *ith loop delay* is the time required for a token to travel around loop *i*.

For each loop, we can define four loop delays depending on the starting point (corresponding to the point *1*, *2*, *3* and *4* in Figure 4.2) of the loop. Among these loop delays, the two starting with points *1* and *2* are the same for fixed stage–delay micropipelines. Similarly, loop delays starting with points *3* and *4* have the same value. Since, as will be seen later, we are interested only in loop delays with loops starting with points *1* and *3*, therefore, these two loop delays will be described as follows for stage *i*, $1 \le i \le m$.

$$\begin{aligned} T_i^1(j+1) &= F_i + D_{i+1}^{24}(j+1) + B_i + D_i^{42}(j+2) \\ &= D_i + D_{i+1}^{24}(j+1) + D_i^{42}(j+2) \\ T_i^3(j+2) &= B_i + D_i^{42}(j+2) + F_i + D_{i+1}^{24}(j+2) \\ &= D_i + D_i^{42}(j+2) + D_{i+1}^{24}(j+2),\ j \ge 0. \end{aligned}$$

Definition 2: We define the *output loop delay* (or *cycle time*) of a m–stage linear micropipeline as the time for a token to travel around loop *m+1*, as shown in Figure 4.1. It is denoted as $T_{m+1}^1(j+1)$, and $T_{m+1}^1(j+1) = D_{out} + D_{m+1}^{42}(j+2)$.

Definition 3: (*Environmental*) *input delay,* D_{in}, is defined as the time required for the environment to prepare and inject new data into a micropipeline. (*Environmental*) *output delay,* D_{out}, is defined as the time required for the environment to collect and process the resulting data from a micropipeline. These delays are indicated pictorially in Figure 4.1. In this chapter, D_{in} and D_{out} are assumed to be constants.

Definition 4: The *m*–stage pipeline performance, *throughput P(j+1),* is defined as the reciprocal of output loop delay $T_{m+1}^1(j+1)$. That is, $P(j+1) = 1/T_{m+1}^1(j+1) = 1/(D_{out} + D_{m+1}^{42}(j+2))$.

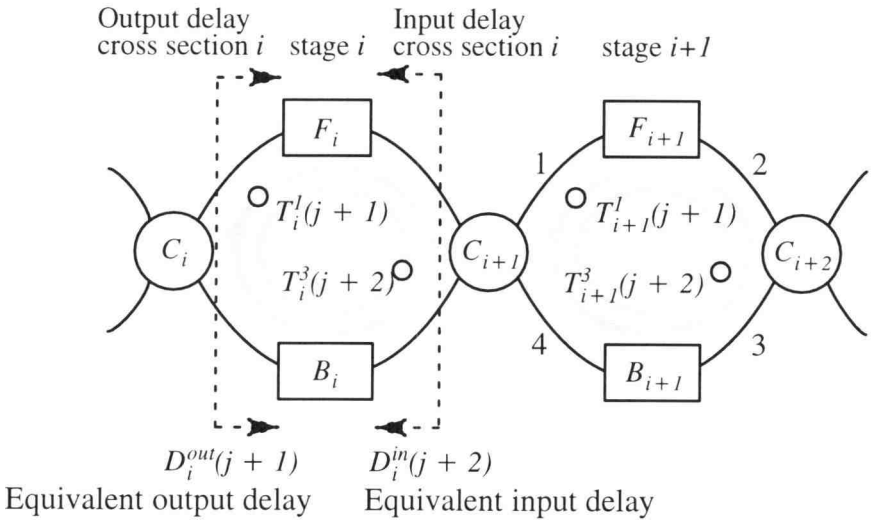


Figure 4.2: Definitions of equivalent input/output delays and loop delays.

Theorem 1: *(Equal loop–delay theorem.)* Looking at C–element C_{i+1} in Figure 4.2, we find that $T_i^3(j+2) = T_{i+1}^1(j+1)$ for all js. This theorem can be applied to all C–elements in a micropipeline.

Let us assume the C–element C_{i+1} has just fired. According to the firing rule of a C–element, the next time it will fire again is equal to the time that each token travels around its own loop. The equation modeling the firing rule for C_{i+1} can be expressed as follows.

$$B_i + D_i^{42}(j+2) + F_i + D_{i+1}^{24}(j+2)$$

$$= F_{i+1} + D_{i+2}^{24}(j+1) + B_{i+1} + D_{i+1}^{42}(j+2), \text{ or equivalently,}$$

$$D_i + D_i^{42}(j+2) + D_{i+1}^{24}(j+2) = D_{i+1} + D_{i+2}^{24}(j+1) + D_{i+1}^{42}(j+2)$$

Several examples for different j are shown in Figure 4.3. The highlighting (bold) in Figure 4.3 indicates the previous equation in pictorial form when $j=k$. Since $D_{i+1}^{24}(j+2)$ and $D_{i+1}^{42}(j+2)$ are "mutually exclusive" (if one is nonzero, the other one must be zero) of each other, in general, we may treat these two delays as one unknown variable. Unfortunately, we still cannot solve the above equation unless both $D_{i+2}^{24}(j+1)$ and $D_i^{42}(j+2)$ are known. This leads to the data dependency graph for the entire micropipeline, as shown in Figure 4.4. With this data dependency graph, we can easily trace and calculate each logic delay in a linear micropipeline. The following analytical approach gives us more insight onto linear micropipelines.

Definition 5: The data token *stage–travelling time* $L_i^{22}(j+1, j+1)$ is defined as the data token travelling from stage i–1 to stage i, as shown in Figure 4.3, and is equal to $D_i^{24}(j+1) + F_i$, $j \geq 0$. Again for the sake of simplicity, $L_i^{22}(j+1, j+1)$ is short–handed to $L_i^{22}(j+1)$.

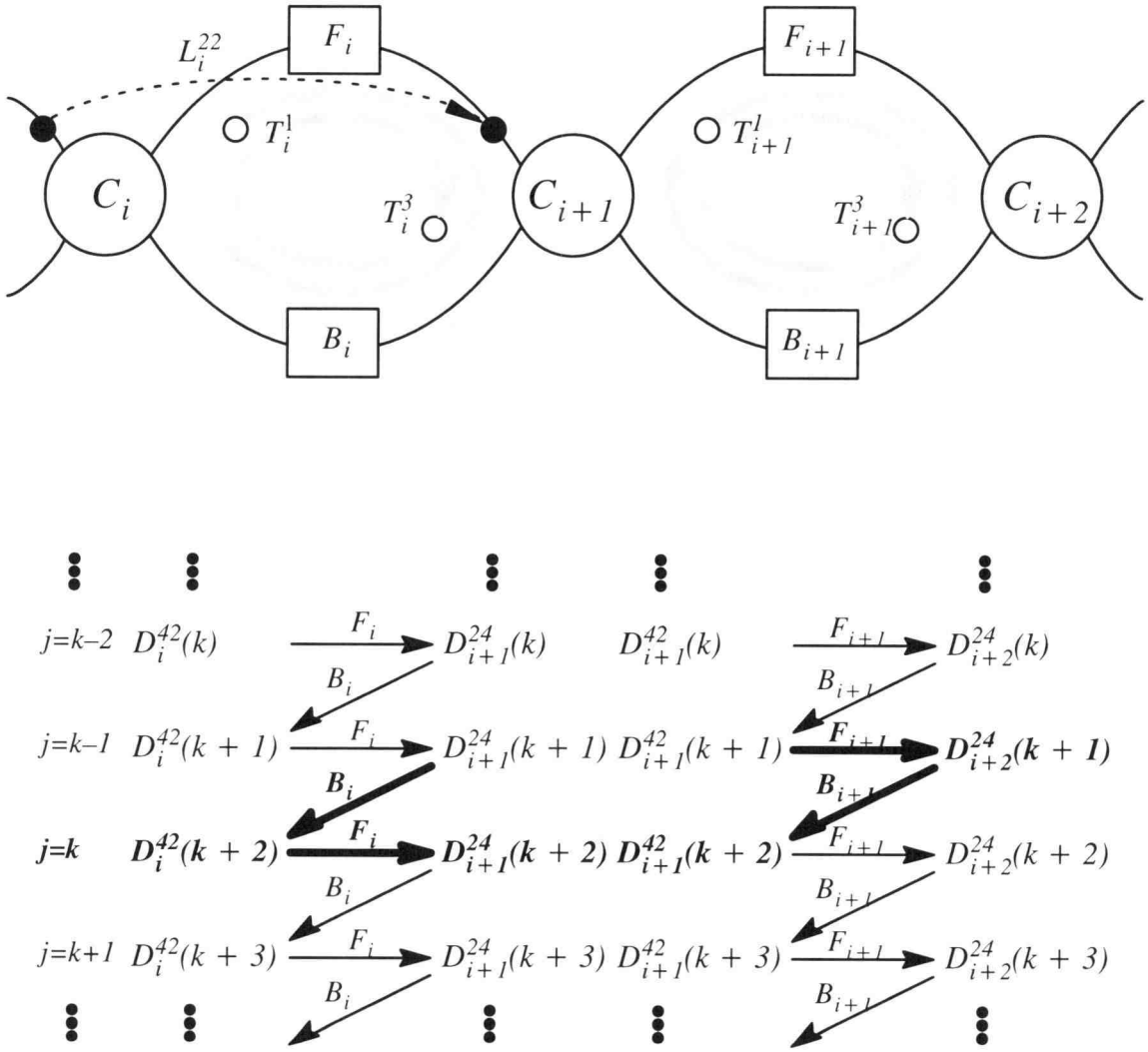


Figure 4.3: Pictorial representation of a firing sequence.

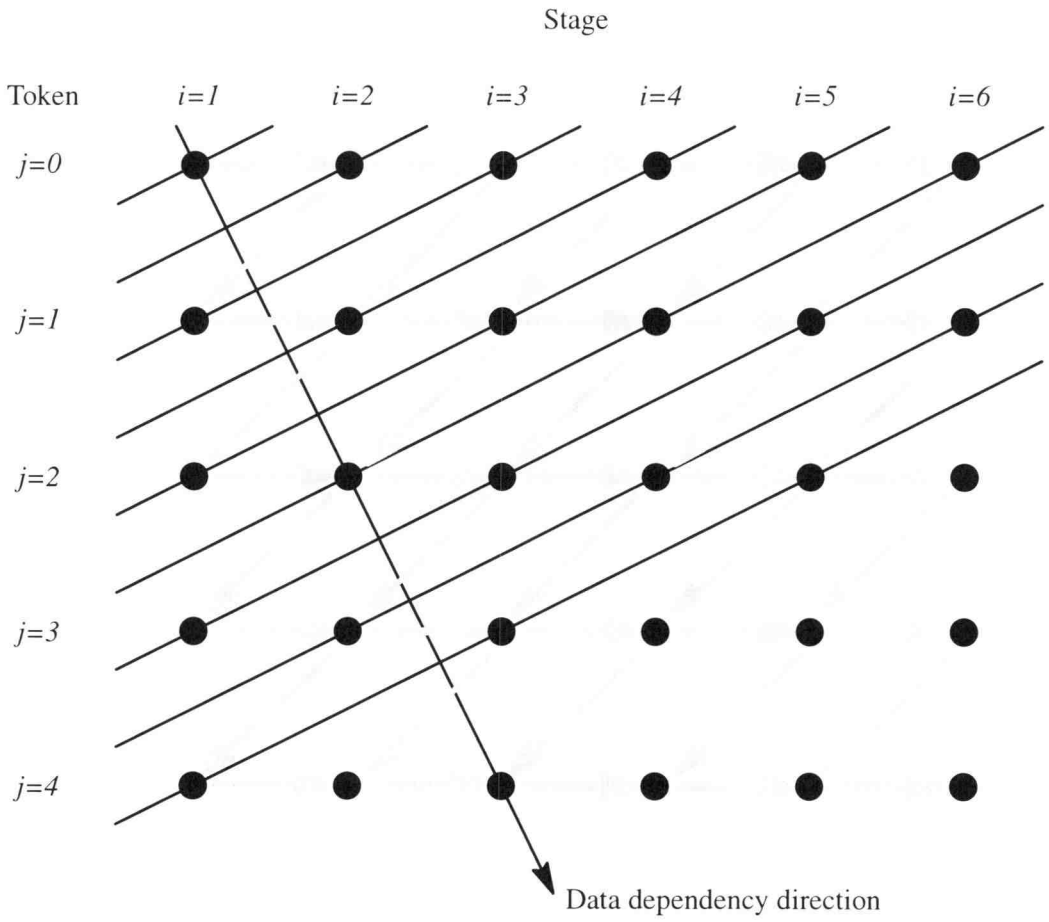


Figure 4.4: A data dependency graph for a linear micropipeline.

Definition 6: The time for a micropipeline to complete its first result is called *latency*, denoted as L. Therefore, a m–stage linear micropipeline has a latency of $L = \sum_{i=1}^{m} L_i^{22}(1)$ $= \sum_{i=1}^{m}(D_i^{24}(1) + F_i) = \sum_{i=1}^{m} F_i$.

Definition 7: (Refer to Figure 4.2) *Equivalent input delay,* $D_i^{in}(j + 2)$, is defined as the delay which is seen by looking into the right cross section of stage i (input delay cross section i). *Equivalent output delay,* $D_i^{out}(j + 1)$, is defined as the delay that is detected by looking into the left cross section of stage i (output delay cross section i). Their values are,

$$D_i^{in}(j + 2) = D_i^{42}(j + 2) + D_i \text{ ,and}$$

$$D_i^{out}(j + 1) = D_{i+1}^{24}(j + 1) + D_i$$

Once the equivalent input delay is defined, all stages to the left of an input delay cross section can be replaced by a stage with its delay value equal to an equivalent input delay. For the same reason, all the stages to the right of an output delay crosss section can be replaced by a stage with its delay value equal to an equivalent output delay. As a result, Figure 4.5(a), (b) and (c) are equivalent.

Theorem 2: Given a m–stage linear micropipeline, an equivalent input delay for each stage i is bounded by $D_i \le D_i^{in}(j + 2) \le Max(D_{in}, D_1.., D_i)$ and a logic delay $D_k^{42}(j + 2)$ is bounded by $0 \le D_k^{42}(j + 2) \le Max(0, D_{k-1}^{in}(j + 2) - D_k)$, where $1 \le i \le m$, $1 \le k \le m + 1$, $D_0^{in}(j + 2) = D_{in}$ and $D_{m+1} = D_{out}$. This is true for all $j \ge 0$.

<Proof> Consider C_1 in Figure 4.5(a). According to Theorem 1,

$D_{in} + D_1^{24}(j + 2) = D_1 + D_2^{24}(j + 1) + D_1^{42}(j + 2)$

If $D_{in} \ge D_1 + D_2^{24}(j + 1)$, then

$$D_1^{24}(j + 2) = 0 \text{ and}$$

$$D_1^{42}(j + 2) = D_{in} - D_1 - D_2^{24}(j + 1) \qquad (4.1)$$

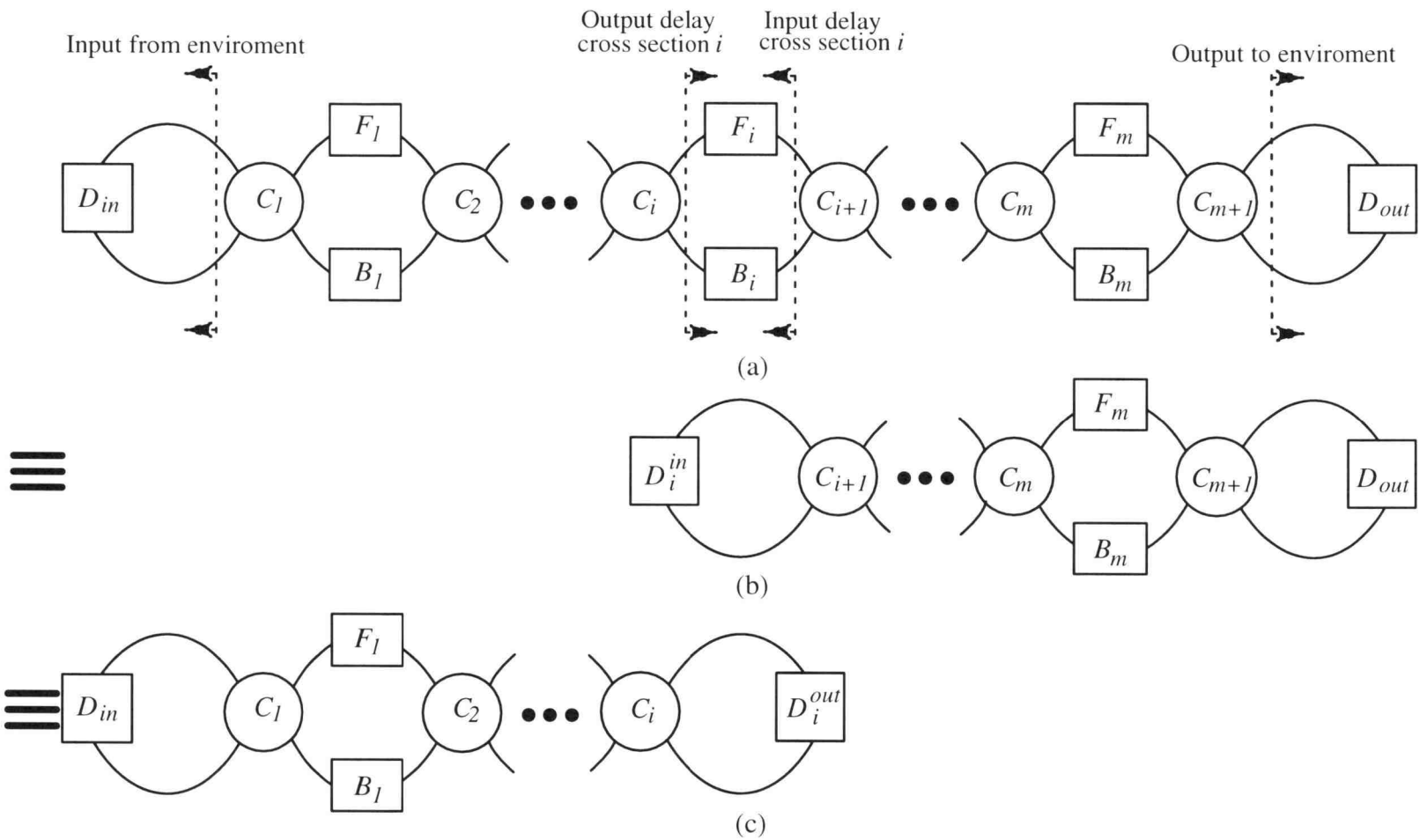


Figure 4.5: A m–stage linear micropipeline and its equivalents.
(a) A linear micropipeline.
(b) First equivalent form.
(c) Second equivalent form.

$$\Rightarrow D_1^{in}(j+2) = D_1 + D_1^{42}(j+2) = D_{in} - D_2^{24}(j+1) \tag{4.2}$$

If $D_{in} \leq D_1 + D_2^{24}(j+1)$, then

$$D_1^{42}(j+2) = 0 \tag{4.3}$$

$$\Rightarrow D_1^{in}(j+2) = D_1 + D_1^{42}(j+2) = D_1 \tag{4.4}$$

From Eq. (4.1) and (4.3),

$$\Rightarrow D_1^{42}(j+2) = 0, \text{ or} \tag{4.5}$$

$$= D_{in} - D_1 - D_2^{24}(j+1) \leq D_{in} - D_1 \tag{4.6}$$

From Eq. (4.2) and (4.4),

$$\Rightarrow D_1^{in}(j+2) = D_{in} - D_2^{24}(j+1) \leq D_{in}, \text{ or} \tag{4.7}$$

$$= D_1 \tag{4.8}$$

Consider C_2 in Figure 4.5(a). According to Theorem 1,

$$D_1^{in}(j+2) + D_2^{24}(j+2) = D_2 + D_3^{24}(j+1) + D_2^{42}(j+2)$$

If $D_1^{in}(j+2) \geq D_2 + D_3^{24}(j+1)$, then

$$D_2^{24}(j+2) = 0, \text{ and}$$

$$D_2^{42}(j+2) = D_1^{in}(j+2) - D_2 - D_3^{24}(j+1) \tag{4.9}$$

$$\Rightarrow D_2^{in}(j+2) = D_2 + D_2^{42}(j+2) = D_1^{in}(j+2) - D_3^{24}(j+1) \tag{4.10}$$

If $D_1^{in}(j+2) \leq D_2 + D_3^{24}(j+1)$, then

$$D_2^{42}(j+2) = 0 \tag{4.11}$$

$$\Rightarrow D_2^{in}(j+2) = D_2 + D_2^{42}(j+2) = D_2 \tag{4.12}$$

From Eq. (4.9) and (4.11),

$$\Rightarrow D_2^{42}(j+2) = 0, \text{ or} \tag{4.13}$$

$$= D_1^{in}(j+2) - D_2 - D_3^{24}(j+1) \leq D_1^{in}(j+2) - D_2 \tag{4.14}$$

From Eq. (4.7), (4.8), (4.10) and (4.12),

$$\Rightarrow D_2^{in}(j+2) = D_{in} - D_2^{24}(j+1) - D_3^{24}(j+1) \leq D_{in}, \text{ or} \tag{4.15}$$

$$= D_1 - D_3^{24}(j+1) \leq D_1, \text{ or} \tag{4.16}$$

$$= D_2 \tag{4.17}$$

The conclusion can be reached by applying an approach similar to that shown above to all C–elements in a micropipeline. □

Corollary 2.1: If $Max(D_{in}, D_1 \ldots D_i) = D_n$, where $1 \leq i \leq m$ and $n \leq i$, then $D_n^{42}(j+2) = 0$ for all js.

<Proof> From Theorem 2, we know

$$D_n^{in}(j+2) \leq Max(D_{in}, D_1 \ldots D_n) = D_n \tag{4.18}$$

Also, from definition

$$D_n^{in}(j+2) = D_n + D_n^{42}(j+2) \tag{4.19}$$

From (4.18) and (4.19), we conclude $D_n^{42}(j+2) = 0$ for all js. □

Theorem 3: Given a m–stage linear micropipeline, equivalent output delay for each stage is bounded by $D_i \leq D_i^{out}(j+1) \leq Max(D_{out}, D_m, \ldots, D_i)$ and logic delay $D_k^{24}(j+2)$ is bounded by $0 \leq D_k^{24}(j+2) \leq Max(0, D_k^{out}(j+1) - D_{k-1})$, where $1 \leq i \leq m$, $1 \leq k \leq m+1$, $D_{m+1}^{out}(j+1) = D_{out}$ and $D_0 = D_{in}$. This is true for all $j \geq 0$.

Corollary 3.1: If $Max(D_i \ldots D_m, D_{out}) = D_n$, where $1 \leq i \leq m$ and $n \geq i$, then $D_{n+1}^{24}(j+1) = 0$ for all js.

Theorem 3 and Corollary 3.1 can be reached using approaches similar to those used in Theorem 2 and Corollary 2.1.

Theorem 4: If $D_{i-1}^{in}(j+2) \geq D_i$ and $D_{i-1}^{in}(j+2) \geq D_{i+1}^{out}(j+1)$ for all the j, then $D_i^{24}(j+1) = 0$, $D_{i+1}^{24}(j+1) = 0$ and $D_i^{42}(j+2) = D_{i-1}^{in}(j+2) - D_i$.

<Proof> In Figure 4.6(a), consider C–element C_i,

$$D_{i-1}^{in}(j+2) + D_i^{24}(j+2) = D_i + D_{i+1}^{24}(j+1) + D_i^{42}(j+2) \tag{4.20}$$

Also, consider C–element C_{i+1},

$$D_i + D_i^{42}(j+2) + D_{i+1}^{24}(j+2) = D_{i+1}^{out}(j+1) + D_{i+1}^{42}(j+2) \tag{4.21}$$

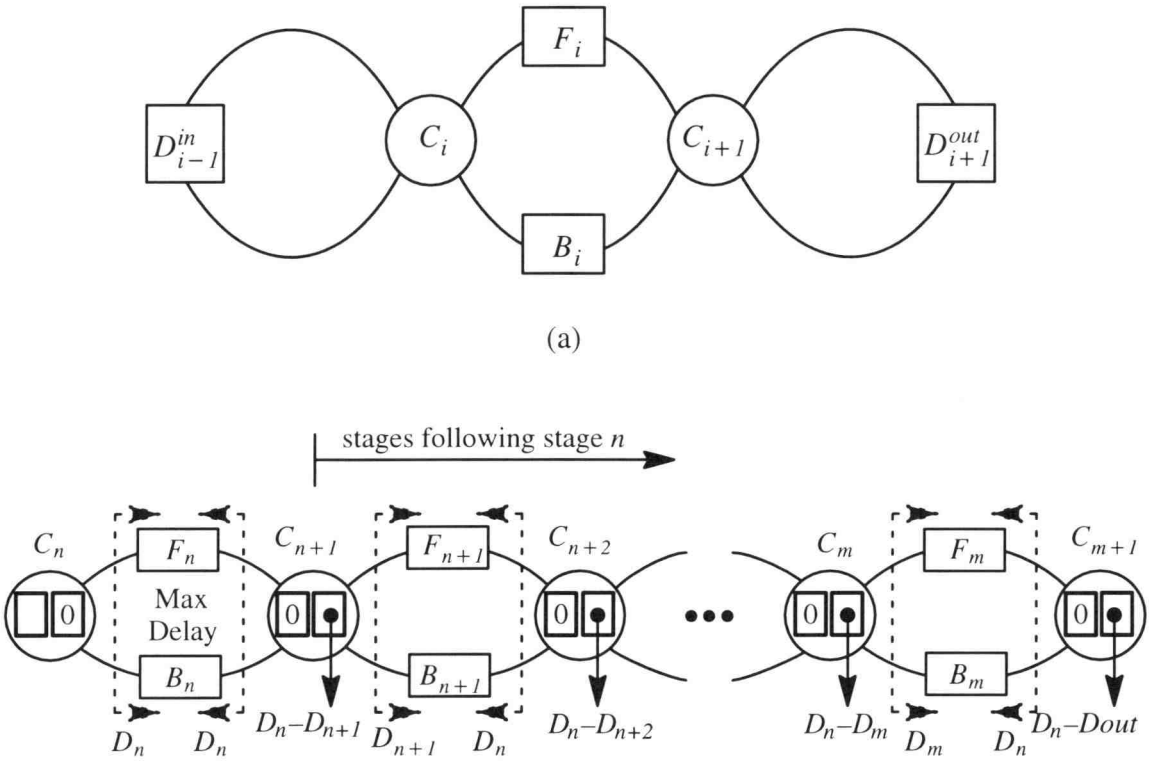


Figure 4.6: Logic and equivalent input/output delay values for the stages following max delay stage n when they become stable.
(a) A general three–stage configuration.
(b) Logic and equivalent input/output delay values for the stages following stage n.

If we add Eq.(4.20) and Eq.(4.21), a new equation is formed as follows.

$$D^{in}_{i-1}(j+2) + D^{24}_{i}(j+2) + D^{24}_{i+1}(j+2)$$
$$= D^{24}_{i+1}(j+1) + D^{out}_{i+1}(j+1) + D^{42}_{i+1}(j+2) \quad (4.22)$$

Because $D^{in}_{i-1}(j+2) \geq D^{out}_{i+1}(j+1)$ for all js, from Eq.(4.22)

$j = 0 \Rightarrow D^{24}_{i+1}(1) = 0 \Rightarrow D^{24}_{i+1}(2) = 0$ to balance Eq.(4.22) on both sides.

$j = 1 \Rightarrow D^{24}_{i+1}(2) = 0 \Rightarrow D^{24}_{i+1}(3) = 0$ to balance Eq.(4.22) on both sides.

If we assume $j=k$ holds, that is

$j = k \Rightarrow D^{24}_{i+1}(k+1) = 0 \Rightarrow D^{24}_{i+1}(k+2) = 0$ to balance Eq.(4.22) on both sides.

When $j=k+1$, because $D^{24}_{i+1}(k+2) = 0$,

$\Rightarrow D^{24}_{i+1}(k+3) = 0$ to balance Eq.(4.22) on both sides.

By induction, $D^{24}_{i+1}(j+1) = 0$ for all js.

From Eq.(4.20), because $D^{24}_{i+1}(j+1) = 0$ and $D^{in}_{i-1}(j+2) \geq D_i$ for all js,

$\Rightarrow D^{24}_{i}(j+1) = 0$ and $D^{42}_{i}(j+2) = D^{in}_{i-1}(j+2) - D_i$. □

Corollary 4.1: If $Max(D_{in}, D_1, \ldots D_m, D_{out}) = D_n$, then $D^{42}_{k}(j+2) = D_n - D_k$ and $D^{24}_{k}(j+1) = 0$, where $n+1 \leq k \leq m+1$ and $j \geq 0$. We define $D_{m+1} = D_{out}$.

<Proof> From Corollary 2.1, $D^{in}_{n}(j+2) = D_n$ and from Theorem 3, $D^{out}_{n+2}(j+1)$ is bounded by $Max(D_{out}, D_m, \ldots, D_{n+2})$, which is less than D_n; therefore, by applying Theorem 4 to all the stages, the above claim can be justified. □

Corollary 4.2: If $Max(D_{in}, D_1, \ldots D_m, D_{out}) = D_n$, then $D^{in}_{n}(j+2) = D^{in}_{n+1}(j+2) = \ldots = D^{in}_{m}(j+2) = D_n$ for all $j \geq 0$.

Corollary 4.3: A m–stage linear micropipeline has throughput $P(j+1)$ equal to the maximum total stage–delay. That is, $P(j+1) = 1/(Max(D_{in}, D_1, \ldots D_m, D_{out}))$.

<Proof> If $Max(D_{in}, D_1, \ldots D_m, D_{out}) = D_n$, by definition, $P(j+1) = 1/T^{1}_{m+1}(j+1) = 1/(D_{out} + D^{42}_{m+1}(j+2))$ and from Corollary 4.1 $D^{42}_{m+1}(j+2) = D_n - D_{m+1} =$

$D_n - D_{out}$; therefore, $P(j+1) = 1/(D_{out} + D^{42}_{m+1}(j+2)) = 1/D_n$. □

Definition 8: A stage k in a linear micropipeline becomes *stable* when both logic delays $D^{42}_k(j+2)$ and $D^{24}_{k+1}(j+1)$ become stable. A logic delay becomes *stable* or reaches *steady state* when its value becomes constant after certain time (token) $j \geq q$.

Note that, according to the Corollary 4.3, throughput is not a function of token index j. It is a constant and equal to the reciprocal of maximum total stage–delay within a whole linear micropipeline. This is not unexpected since from Corollary 4.1, we know that all logic delays following the stage with maximum stage–delay are always constant. The above discussion is summarized and shown in Figure 4.6(b) when all stages in a micropipeline have reached steady state. In Figure 4.6(b), logic delays $D^{24}_i(j+1) = 0$ and $D^{42}_i(j+2) = D_n - D_i$, where i refers to all stages following the maximum delay stage n.

Up to this point, all logic delays following the stage with maximum total stage–delay have been solved as shown previously. If we can also solve the logic delays prior to the stage with maximum total stage–delay, the whole micropipeline is then fully determined. That is, the traverse of token flow for each j will become solvable. The following definitions and theorems will help us lay the ground for the transient and steady state analysis of logic delays discussed in the next section.

Theorem 5: If $D^{in}_{i-1}(j+2) \leq D_i$ for all $j \geq 0$, then $D^{42}_i(j+2) = 0$, where $1 \leq i \leq m+1$. We define $D^{in}_0(j+2) = D_{in}$ and $D_{m+1} = D_{out}$.

This theorem is obvious as long as the governing equation based upon Equal loop–delay theorem is written out.

Definition 9: Given a m–stage linear micropipeline (refer to Figure 4.7),

if $Max(D_{in}, D_1, ..., D_m, D_{out}) = D_n$, then D_n is called the *1st local maximum delay*;

if $Max(D_{in}, D_1, ..., D_{n-1}) = D_q$, then D_q is called the *2nd local maximum delay*;

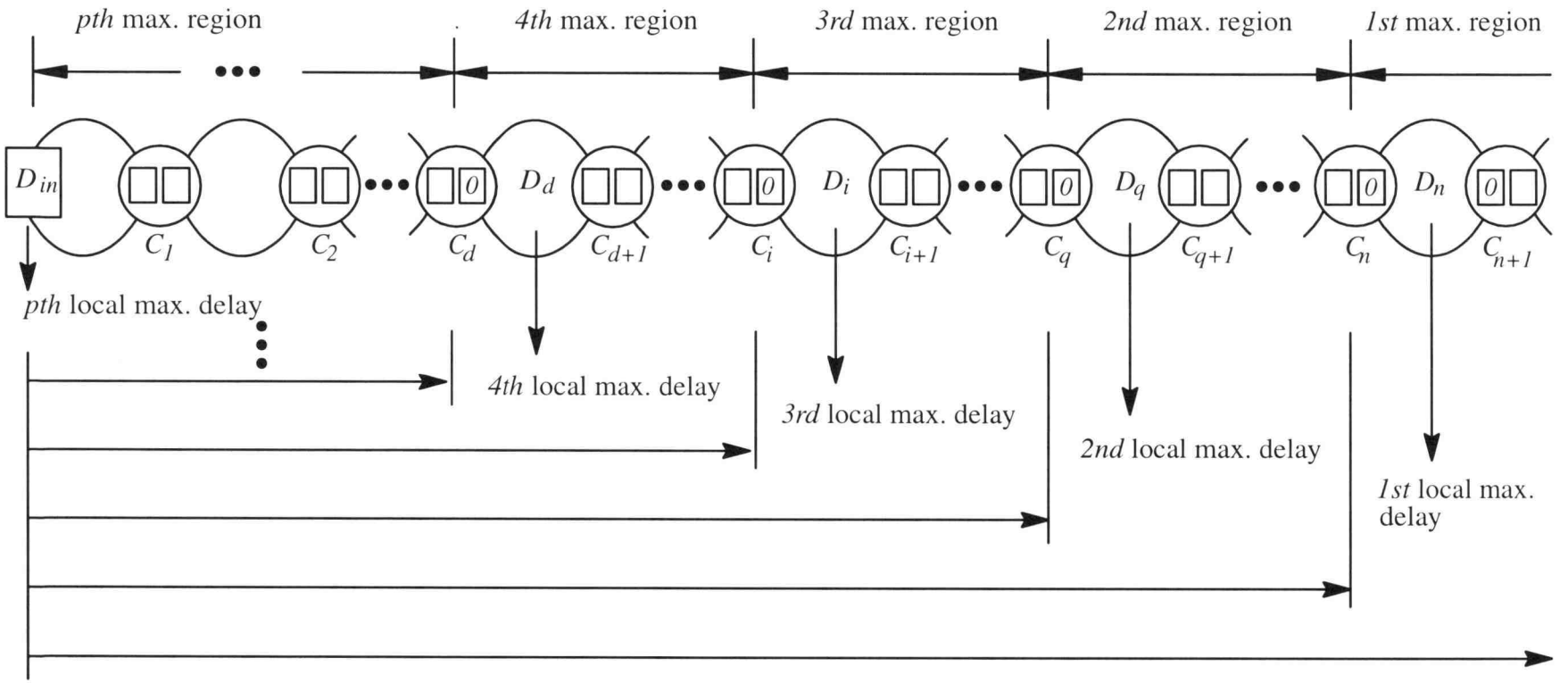


Figure 4.7: Definitions for local maximum delays and maximum regions.

if $Max(D_{in}, D_1, ..., D_{q-1}) = D_i$, then D_i is called the *3rd local maximum delay.*

The naming process continues until D_{in} becomes the *pth local maximum delay.*

It should be noted that, in general, the *rth* local maximum delay may not be equal to the *rth* global maximum delay for $2 \leq r \leq p$. The *rth* local maximum delay is defined as the maximum delay among the ones corresponding to the left–hand–side stages of the *(r–1)th* local maximum delay. However, the *rth* global maximum delay is always defined over the whole micropipeline. We define the *1st* local maximum delay as equal to the *1st* global maximum delay. For convenience, they are called maximum delay or maximum stage–delay.

Definition 10: The *sth maximum region*, as shown in Figure 4.7, is the region containing stages between *sth* local maximum delay stage (including) and *(s–1)th* local maximum delay stage (excluding), where $1 \leq s \leq p$. The *0th* local maximum delay stage is defined as the maximum–delay stage.

Corollary 5.1: If r represents the stages having local maximum delays, then $D_r^{42}(j + 2) = 0$ for all $j \geq 0$.

This corollary can be derived from Theorem 2 and Theorem 5. The result is shown in Figure 4.7.

Corollary 5.2: If stage s corresponds to the *bth* local maximum delay stage, then $D_i^{42}(2) = D_s - D_i$, where i represents the stages within the *bth* maximum region.

Corollary 5.3: If $D_i^{in}(j + 2) \geq D_{i+1}^{out}(j + 1)$ for $j=0$, then $D_{i+1}^{24}(j + 2)$ will remain zero until $D_i^{in}(j + 2) \leq D_{i+1}^{out}(j + 1)$ for $j \geq k$.

According to the firing rule and Corollary 5.2 and 5.3, $D_i^{in}(j + 2)$ will not change until $D_i^{in}(j + 2) \leq D_{i+1}^{out}(j + 1)$ for $j \geq k$.

4.2 Transient Delay And Steady State Analysis

With the definitions and theorems provided in the previous section, we are ready to further explore the transient and steady state analysis of logic delays for the whole micropipeline. Given a *m*–stage linear micropipeline, assume that the *nth* stage has the *1st* global maximum delay, D_n, and the *ith* stage has the *2nd* local maximum delay, D_i. Figure 4.8(a) shows the circuit for the transient delay and steady state analysis of the *(n–1)th* stage. According to the firing rule and Corollary 5.2 and 5.3, $D^{in}_{n-2}(j+2)$ will remain equal to D_i until $D^{in}_{n-2}(j+2) \le D^{out}_{n-1}(j+1)$ for $j \ge k$. This is the situation when stage *n–1* becomes stable. Looking at Figure 4.8(b), because we assume stage *n–1* turns out to be stable at $j = k + 1$, it implies that

$$k(D_n - D_i) + D_{n-1} \ge D_i$$

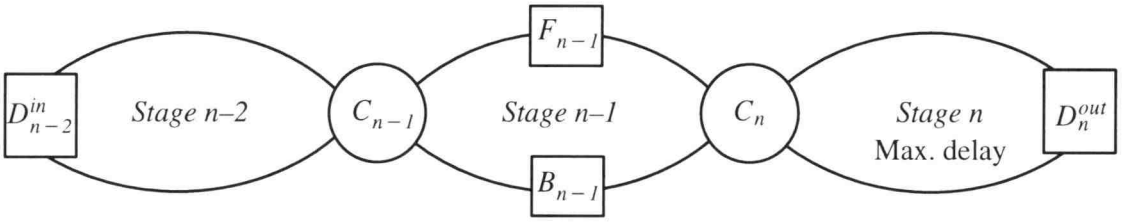
$$\Rightarrow k \ge \frac{D_i - D_{n-1}}{D_n - D_i}, \tag{4.23}$$

where k is the minimum integer satisfying this equation.

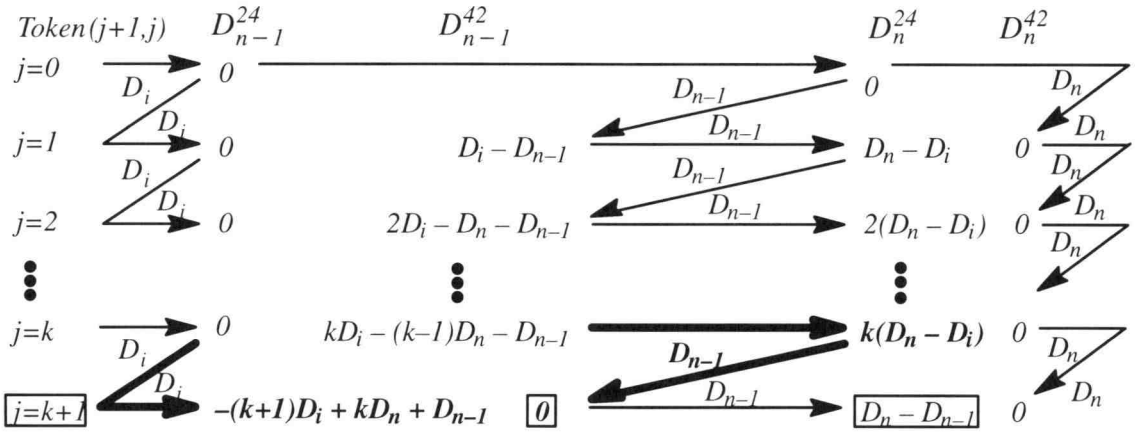
$$D^{24}_n(j+1) = \begin{cases} 0 & \text{, for } 0 \le j \le 0 \\ j(D_n - D_i) & \text{, for } 1 \le j \le k \\ D_n - D_{n-1} & \text{, for } k+1 \le j \end{cases} \tag{4.24}$$

$$D^{42}_{n-1}(j+1) = \begin{cases} D_i - D_{n-1} & \text{, for } 1 \le j \le 1 \\ jD_i - (j-1)D_n - D_{n-1} & \text{, for } 2 \le j \le k \\ 0 & \text{, for } k+1 \le j \end{cases} \tag{4.25}$$

Delay $D^{24}_n(j+1)$ in Eq.(4.24) is considered first. For $1 \le j \le k$, $D^{24}_n(j+1)$ increases as j increases because $(D_n - D_i) > 0$. Is $D_n - D_{n-1}$ for $j \ge k+1$ greater than $j(D_n - D_i)$ at j=k? From Eq.(4.23), assume $D_i - D_{n-1} = Q(D_n - D_i) + R$, i.e. $Q = (D_i - D_{n-1} - R)/(D_n - D_i)$, where Q is integer and $0 \le R < D_n - D_i$. Since k is the minimum integer satisfying Eq.(4.23), let $j = k = Q + 1$ for the worst case (the situation when $R > 0$). Therefore, $D^{24}_n(j+1) = (Q+1)(D_n - D_i) = D_n - D_{n-1} - R$ at $j = k < D_n - D_{n-1} = D^{24}_n(j+1)$ at $j = k + 1$. From the above discussion, it is



(a)



(b)

Figure 4.8: Transient delay and steady state analyses.
(a) Circuit .
(b) Equation derivation.

summarized that $D_n^{24}(j+1)$ increases with j and when it reaches $D_n - D_{n-1}$, it would no longer change (becoming stable).

We then consider delay $D_{n-1}^{42}(j+1)$ in Eq.(4.25). For $j=1$, $D_{n-1}^{42}(j+1) = D_i - D_{n-1}$. For the second case $2 \le j \le k$, $D_{n-1}^{42}(j+1) = jD_i - (j-1)D_n - D_{n-1} = (D_i - D_{n-1}) + (j-1)(D_i - D_n)$. Because $(D_i - D_n) < 0$, $D_{n-1}^{42}(j+1)$ becomes smaller when j becomes larger. In addition, $D_{n-1}^{42}(j+1)$ could not be negative; therefore, we conclude that $D_{n-1}^{42}(j+1)$ decreases when j increases. It eventually reaches zero and will no longer change (reaching steady state).

The above equations (4.23), (4.24) and (4.25) are valid only for the stage *n–1* provided that stage *n* has the maximum stage–delay. However, using a similar approach, they can be generalized to apply to all stages within the *2nd* maximum region. As a result, a more general formula can be obtained for these stages. Again, if maximum delay is D_n and *2nd* local maximum delay is D_i, then the equations for transient delays and steady state analysis of the *(n–q)th* stage, where $i \le n-q \le n-1$ or $1 \le q \le (n-i)$ are

$$k_q \ge \frac{D_i + (q-1)D_n - \sum_{r=1}^{q} D_{n-r}}{D_n - D_i} \tag{4.26}$$

where k_q is the minimum integer satisfying this equation.

$$D_{n-q+1}^{24}(j+1) = \begin{cases} 0 & \text{, for } 0 \le j \le k_{q-1} \\ -jD_i + (j-q+1)D_n + \sum_{r=1}^{q-1} D_{n-r} & \text{, for } k_{q-1}+1 \le j \le k_q \\ D_n - D_{n-q} & \text{, for } k_q + 1 \le j \end{cases} \tag{4.27}$$

$$D^{42}_{n-q}(j+1) = \begin{cases} D_i - D_{n-q} & \text{, for } 1 \le j \le k_{q-1} + 1 \\ jD_i - (j-q)D_n - \sum_{r=1}^{q} D_{n-r} & \text{, for } k_{q-1} + 2 \le j \le k_q \\ 0 & \text{, for } k_q + 1 \le j \end{cases} \quad (4.28)$$

where $k_0=0$ and $k_{n-i}=k_{n-(i+1)} + 1$.

Figure 4.9, which has gone through several time steps, illustrates part of a linear micropipeline with a maximum delay of 17 and a *2nd* local maximum delay of 15. The values of transient delays for each logic delay in Figure 4.9 can be easily verified with Eq.(4.26), (4.27) and (4.28) given $D_n = 17$, $D_{n-1} = 12$, $D_{n-2} = 10$, $D_{n-3} = D_i = 15$ and $1 \le q \le 3$. Substituting these values into Eq.(4.26), we get $k_1 = 2$, $k_2 = 5$ and $k_3 = 6$. In other words, stage *n–1* reaches steady state at *j=3*, stage *n–2* at *j=6* and stage *n–3* at *j=7*. The values corresponding to steady state are enclosed in a box in Figure 4.9.

Note that $D^{24}_{n-q}(j + 1)$ increasing with j and $D^{42}_{n-q}(j + 1)$ decreasing with j are true for all stages within the *2nd* maximum region (actually, this statement is valid for all stages, given a linear micropipeline). Before stage *n–1* reaches steady state, none of the logic delays for stages prior to stage *n–1,* but within the *2nd* maximum region, change. At the moment stage *n–1* reaches stable state, stage *n–2* starts to change by increasing $D^{24}_{n-1}(j + 1)$ and decreasing $D^{42}_{n-2}(j + 1)$. This domino process continues until stage *i*, the *2nd* local maximum delay stage, reaches its steady state.

In reality, the Eq.(4.26) ,(4.27) and (4.28) hold for ***all*** the maximum regions within a linear micropipeline before $D^{24}_{t+1}(j + 1)$ starts change, where t corresponds to all the local maximum delay stages. For example, in the *3rd* maximum region, D_i in Eq.(4.26) ,(4.27) and (4.28) refers to the *3rd* local maximum delay, D_n represents the *2nd* local maximum delay and D_{n-r} are the delays corresponding to the stages within this region. All stages within maximum regions attempt to reach steady state simultaneously. However, except for stages within *2nd* maximum region, all other stages reach a temporary steady state only by following Eq.(4.26) ,(4.27) and (4.28). The reason for

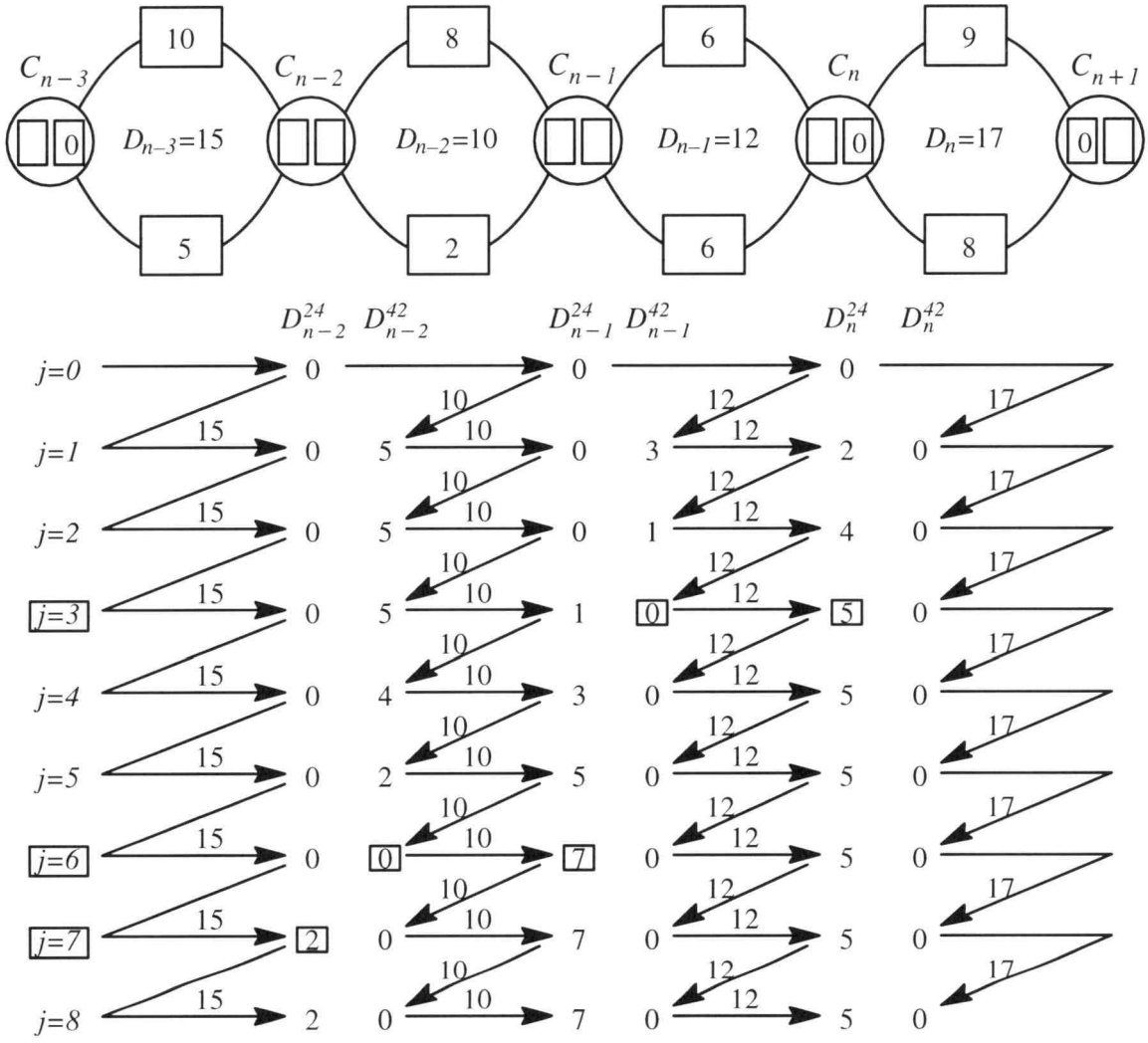


Figure 4.9: Numerical example for a four–stage linear micropipeline.

this is that when the Eq.(4.23), (4.24) and (4.25) were derived, $D^{24}_{n+1}(j+1) = 0$ for all $j \geq 0$; therefore, $D^{out}_{n}(j+1) = D_n + D^{24}_{n+1}(j+1) = D_n$. However, for example, in the *3rd* maximum region, assuming that D_i is equal to *2nd* local maximum delay, $D^{24}_{i+1}(j+1)$ is not always zero; hence, $D^{out}_{i}(j+1) = D_i + D^{24}_{i+1}(j+1)$ is not always equal to D_i. This is why all stages prior to the *2nd* maximum region reach temporary steady state only when $D^{24}_{n+1}(j+1) = 0$. When $D^{24}_{i+1}(j+1)$ starts to change, as it does when stage *i+1* becomes stable, $D^{out}_{i}(j+1)$ changes also. This change moves the stages away from temporary steady state and toward the final steady state. In terms of timing, stages in the *3rd* maximum region may reach temporary steady state either faster or later than stages in the *2nd* maximum region. This makes the analytic approach more difficult, if not impossible. However, no matter how complex the situation is, all stages prior to the *1st* local maximum delay stage must eventually reach the final steady state with $D^{42}_{r}(j+1) = 0$ for $j \geq 1$ and $D^{24}_{r+1}(j+1) = D_n - D_r$ for $j \geq 0$ and $1 \leq r \leq n-1$, as shown in Figure 4.10.

4.3 Summary

This chapter defines several terms to be used in the following chapters for conveniently describing the operation and performance of micropipelines. Some theorems and corollaries were also developed to target the transient and steady states of logic delays for each C–element (stage) and the performance of micropipelines. For the stages prior to the maximum delay stage (assuming stage n), logic delays $D^{24}_{i+1}(j+1)$ increases and $D^{42}_{i}(j+1)$ decreases when j increases, where $1 \leq i \leq n$. Each logic delay stops change when the steady state of corresponding stages is reached. However, the logic delays $D^{24}_{i+1}(j+1)$ and $D^{42}_{i}(j+1)$ for stages following the maximum delay stage, are constant, where $n+1 \leq i \leq m+1$. As a result, the throughput of a linear micropipeline with

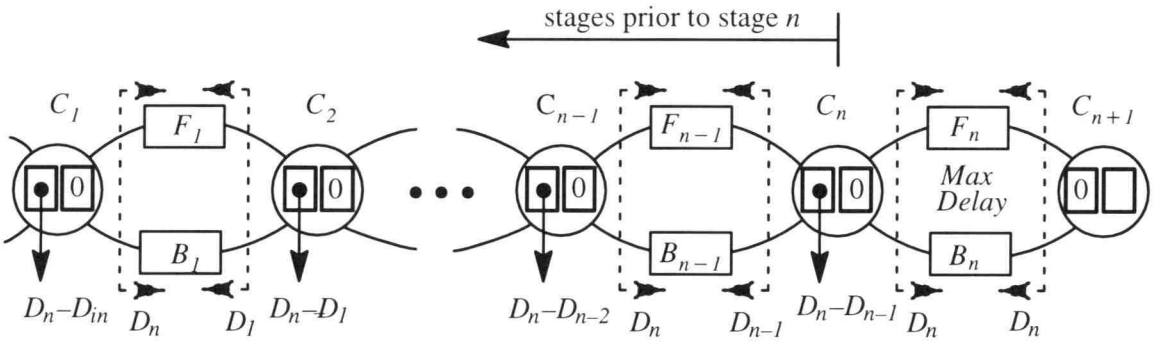


Figure 4.10: Logic and equivalent input/output delay values for the stages prior to stage n when steady state is reached for all stages.

fixed stage–delay is a constant and equals the inverse of the maximum total stage–delay. Giving the difficulty in obtaining analytical solutions to logic delays in closed form for fixed stage–delay micropipelines, the possibility of doing so for variable stage–delay micropipelines is inconceivable. Therefore, we will not find the average throughput for variable stage–delay micropipelines; instead, we are more interested in the throughput bounds, which will be discussed in the following chapter.

5. THE PERFORMANCE OF A LINEAR MICROPIPELINE WITH VARIABLE STAGE–DELAY

In this chapter, the properties of a linear micropipeline with variable stage–delays are discussed. A variable stage–delay comes from the completion–detection technique [6,7,8] used for encoding the data in each stage. As noted in the previous chapter, finding out analytical solutions to logic delays for fixed stage–delay micropipelines is very difficult, if not impossible. More difficulty is expected in doing so for micropipelines with variable stage–delays. This prohibits us from analyzing the average throughput in closed form (we will discuss the average throughput in numerical form in Chapter 6). However, we are able to obtain the throughput bounds. Therefore, we are only interested in the throughput bounds of micropipelines in this chapter. First, various representations of logic delays (and hence output loop–delays) are given. Second, based upon one of these representations, a design procedure is introduced and several design guidelines to achieve required bounds are suggested. Finally, an example is used to demonstrate the design procedure and guidelines, given the constraints of lower and upper bounds of output loop–delay.

5.1 Upper And Lower Performance Bounds

From previous discussions, the performance of a micropipeline with fixed stage–delays is limited by the maximum total stage–delay. Ignoring the overhead of a C–element's physical delay, this is approximately the same as a synchronous pipeline's performance (a more detailed comparison will be discussed in Chapter 6). From the performance point of view, an asynchronous approach provides no benefit over the synchronous approach. This is because a constant delay element which is greater than or equal to the worst–case delay of a combinational logic circuit must be inserted at each stage's data–token path. However, in many cases, the strategy of adding worst–case

delay in each stage is too conservative and cannot take full advantage of the "average" case performance. Hence, completion detection techniques are developed to improve the possible speedup, leading to variable stage–delay for each stage [6,7,8]. That is, stage–delay is variable and is a function of token index j (time). More precisely, the *ith* forward delay, *ith* backward delay, input delay and output delay are represented as $F_i(j+1)$, $B_i(j+1)$, $D_{in}(j+2)$ and $D_{out}(j+1)$, respectively.

In this section, we are interested in obtaining the upper and lower performance bounds of an FIFO. As will be shown later in this thesis, it is also very helpful to know the equivalent input and output delays of each stage. By reviewing our FIFO model, we know that, as long as all of the logic delays $D_i^{24}(j+2)$ ($D_i^{24}(1)=0$) and $D_i^{42}(j+2)$ become known, all of the performance–related issues can be derived. In the following, different logic delay representations and approximations are examined. The approximation error is also depicted.

5.1.1 Several Representations Of Logic Delays $D_i^{24}(j+2)$ And $D_i^{42}(j+2)$

All definitions and Theorem 1 (Equal loop–delay) stated in Chapter 4 are applicable to variable stage–delay micropipelines by replacing all fixed delays with variable delays. As a result, the two different loop delays for loop i in the variable stage–delay version are stating as follows.

$$T_i^1(j+1) = F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) + D_i^{42}(j+2)$$

$$T_i^3(j+2) = B_i(j+1) + D_i^{42}(j+2) + F_i(j+2) + D_{i+1}^{24}(j+2)$$

As mentioned in Chapter 4, given a m–stage linear micropipeline, its throughput is

$$P(j+1) = 1/T_{m+1}^1(j+1) = 1/(D_{out}(j+1) + D_{m+1}^{42}(j+2))$$

Obtaining an analytical solution of throughput in closed form (implying that logic delay $D_{m+1}^{42}(j+2)$ should be solved) involves solving a set of linear/non–linear difference

equations derived from all of the C–elements in a pipeline. Linear difference equations are established according to Theorem 1 stated in Chapter 4. The non–linear difference equations come from C–element modeling, i.e. $D_i^{24}(j+1) * D_i^{42}(j+1) = 0$. Solving linear/nonlinear difference equation set is not an easy task. Therefore, logic delays are retained in the representation of output loop delays. As will be seen in the following chapter, this representation is sufficient to explain some interesting characteristics of micropipelines. However, we can "solve" these linear/non–linear difference equations in numerical form directly by substituting j from 0 to k, where k is any positive integer. Actually, all simulation results to be discussed in Chapter 6 are obtained by solving these difference equations using Matlab [30]. Using an approach similar to that described in the proof of Theorem 2 (Chapter 4), we will prove that output loop delay and logic delays have the following representations.

$$
\begin{aligned}
T_{m+1}^{1}(j+1) = Max(&DF_{m+1}(j+1),\\
&DB_m(j+1),\\
&DB_{m-1}(j+1) + \sum_{l=m}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=m+1}^{m+1} D_l^{24}(j+1),\\
&DB_{m-2}(j+1) + \sum_{l=m-1}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=m}^{m+1} D_l^{24}(j+1),\\
&\cdots\cdots\\
&DB_1(j+1) + \sum_{l=2}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=3}^{m+1} D_l^{24}(j+1),\\
&DB_0(j+1) + \sum_{l=1}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=2}^{m+1} D_l^{24}(j+1)) \qquad (5.1)
\end{aligned}
$$

$$
\begin{aligned}
&D_k^{24}(j+2)\\
&= Max(0,\\
&\quad DF_k(j+1) - DB_{k-1}(j+1) - \sum_{l=k-1}^{k-1} D_l^{42}(j+1+k-l),
\end{aligned}
$$

$$DF_{k+1}(j) + \sum_{l=k}^{k}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{k} D_l^{42}(j+1+k-l),$$

. . . . . .

$$DF_m(j+1-m+k) + \sum_{l=k}^{m-1}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m-1} D_l^{42}(j+1+k-l),$$

$$DF_{m+1}(j-m+k) + \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m} D_l^{42}(j+1+k-l)) \tag{5.2}$$

$D_k^{42}(j+2)$

$= Max(0,$

$$DB_{k-1}(j+1) - DF_k(j+1) - \sum_{l=k+1}^{k+1} D_l^{24}(j+1),$$

$$DB_{k-2}(j+1) + \sum_{l=k-1}^{k-1}(F_l(j+2) - F_l(j+1)) - DF_k(j+1) - \sum_{l=k}^{k+1} D_l^{24}(j+1),$$

. . . . . .

$$DB_1(j+1) + \sum_{l=2}^{k-1}(F_l(j+2) - F_l(j+1)) - DF_k(j+1) - \sum_{l=3}^{k+1} D_l^{24}(j+1),$$

$$DB_0(j+1) + \sum_{l=1}^{k-1}(F_l(j+2) - F_l(j+1)) - DF_k(j+1) - \sum_{l=2}^{k+1} D_l^{24}(j+1)) \tag{5.3}$$

where $1 \le k \le m+1$,

$$DF_i(j+1) = F_i(j+1) + B_i(j+1),$$

$$DB_i(j+1) = F_i(j+2) + B_i(j+1),\ 1 \le i \le m$$

$$DB_0(j+1) = D_{in}(j+2),$$

$$DF_{m+1}(j+1) = D_{out}(j+1),$$

$$D_0^{42}(j+v) = 0,$$

$$D_{m+2}^{24}(j+v) = 0,\ j \geq 0 \text{ and } v \in integer.$$

<Proof> Consider C_1 in Figure 4.5(a). According to Theorem 1 in Chapter 4,

$$D_{in}(j+2) + D_1^{24}(j+2) = DF_1(j+1) + D_2^{24}(j+1) + D_1^{42}(j+2)$$

If $D_{in}(j+2) \leq DF_1(j+1) + D_2^{24}(j+1)$, then

$$D_1^{24}(j+2) = DF_1(j+1) + D_2^{24}(j+1) - D_{in}(j+2), \text{ and} \tag{5.4}$$

$$D_1^{42}(j+2) = 0 \tag{5.5}$$

If $D_{in}(j+2) \geq DF_1(j+1) + D_2^{24}(j+1)$, then

$$D_1^{24}(j+2) = 0, \text{ and} \tag{5.6}$$

$$D_1^{42}(j+2) = D_{in}(j+2) - DF_1(j+1) - D_2^{24}(j+1) \tag{5.7}$$

From the conditions and Eq.(5.4) and (5.6),

$$\Rightarrow D_1^{24}(j+2) = Max(0, DF_1(j+1) + D_2^{24}(j+1) - D_{in}(j+2)) \tag{5.8}$$

From the conditions and Eq.(5.5) and (5.7),

$$\Rightarrow D_1^{42}(j+2) = Max(0, D_{in}(j+2) - DF_1(j+1) - D_2^{24}(j+1)) \tag{5.9}$$

Consider C_2 in Figure 4.5(a). According to Theorem 1,

$$DB_1(j+1) + D_1^{42}(j+2) + D_2^{24}(j+2) = DF_2(j+1) + D_3^{24}(j+1) + D_2^{42}(j+2)$$

If $DB_1(j+1) + D_1^{42}(j+2) \leq DF_2(j+1) + D_3^{24}(j+1)$, then

$$D_2^{24}(j+2) = DF_2(j+1) + D_3^{24}(j+1) - DB_1(j+1) - D_1^{42}(j+2), \text{ and} \tag{5.10}$$

$$D_2^{42}(j+2) = 0 \tag{5.11}$$

If $DB_1(j+1) + D_1^{42}(j+2) \geq DF_2(j+1) + D_3^{24}(j+1)$, then

$$D_2^{24}(j+2) = 0, \text{ and} \tag{5.12}$$

$$D_2^{42}(j+2) = DB_1(j+1) + D_1^{42}(j+2) - DF_2(j+1) - D_3^{24}(j+1) \tag{5.13}$$

From the conditions and Eq.(5.10) and (5.12),

$$\Rightarrow D_2^{24}(j+2) = Max(0, DF_2(j+1) + D_3^{24}(j+1) - DB_1(j+1) - D_1^{42}(j+2)) \quad (5.14)$$

From the conditions and Eq.(5.11) and (5.13),

$$\Rightarrow D_2^{42}(j+2) = Max(0, DB_1(j+1) + D_1^{42}(j+2) - DF_2(j+1) - D_3^{24}(j+1)) \quad (5.15)$$

Replacing $D_1^{42}(j+2)$ in Eq.(5.15) with Eq.(5.9), we have

$$\Rightarrow D_2^{42}(j+2)$$

$$= Max(0,$$

$$DB_1(j+1) - DF_2(j+1) - \sum_{l=3}^{3} D_l^{24}(j+1),$$

$$DB_0(j+1) + \sum_{l=1}^{1}(F_l(j+2) - F_l(j+1)) - DF_2(j+1) - \sum_{l=2}^{3} D_l^{24}(j+1)) \quad (5.16)$$

The general form of logic delay $D_k^{42}(j+2)$, as shown in expression (5.3), can be easily reached by applying similar approach to all of the C–elements in a micropipeline. Since $D_k^{42}(j+2)$ is known ($1 \le k \le m+1$) and $T_{m+1}^{1}(j+1) = D_{out}(j+1) + D_{m+1}^{42}(j+2)$, expression (5.1) can easily be verified. Logic delay $D_k^{24}(j+2)$ in expression (5.2) can also be obtained by backward substitution. For example, substituting $D_2^{24}(j+1)$ in Eq.(5.8) with $D_2^{24}(j+2)$ in Eq.(5.14) (note that, index change is required). □

Note that expression (5.1), (5.2) and (5.3) are not unique representations for output loop delay and logic delays. In the proof of (5.1), (5.2) and (5.3), Equal loop–delay theorem is applied to all C–elements from C_1 to C_{m+1}. If Equal loop–delay theorem is applied to all C–elements from C_{m+1} to C_1 and the substitution is made such that only the same type of logic delays ($D_k^{24}(j+2)$ or $D_k^{42}(j+2)$), but different index, appear in the same expression, the alternative representations are shown as follows.

$$T_{m+1}^{1}(j+1) = Max(DF_{m+1}(j+1),$$

$$DB_m(j+1) + D_m^{42}(j+2)) \quad (5.17)$$

$$D_k^{24}(j+2)$$

$$= Max(0,\ min($$

$$DF_k(j+1) - DB_{k-1}(j+1) + \sum_{l=k+1}^{k+1} D_l^{24}(j+1),$$

$$DF_k(j+1) - \sum_{l=k-1}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_{k-2}(j+1) + \sum_{l=k}^{k+1} D_l^{24}(j+1),$$

$$\cdots\cdots$$

$$DF_k(j+1) - \sum_{l=2}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_1(j+1) + \sum_{l=3}^{k+1} D_l^{24}(j+1),$$

$$DF_k(j+1) - \sum_{l=1}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_0(j+1) + \sum_{l=2}^{k+1} D_l^{24}(j+1))) \quad (5.18)$$

$$D_k^{42}(j+2)$$

$$= Max(0,\ min($$

$$DB_{k-1}(j+1) - DF_k(j+1) + \sum_{l=k-1}^{k-1} D_l^{42}(j+1+k-l),$$

$$DB_{k-1}(j+1) - \sum_{l=k}^{k} (B_l(j+1+k-l) - B_l(j+k-l)) - DF_{k+1}(j)$$

$$+ \sum_{l=k-1}^{k} D_l^{42}(j+1+k-l),$$

$$\cdots\cdots$$

$$DB_{k-1}(j+1) - \sum_{l=k}^{m-1} (B_l(j+1+k-l) - B_l(j+k-l)) - DF_m(j+1-m+k)$$

$$+ \sum_{l=k-1}^{m-1} D_l^{42}(j+1+k-l),$$

$$DB_{k-1}(j+1) - \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_{m+1}(j-m+k)$$

$$+ \sum_{l=k-1}^{m} D_l^{42}(j+1+k-l))) \tag{5.19}$$

It is not surprising to know that expressions (5.2) and (5.19) have similar forms, except for having opposite signs and the disparity of *min* and *Max* when the previous proof is reviewed (e.g., comparing Eq. (5.8) and (5.9)). A similar argument can apply to expressions (5.3) and (5.18). Expression (5.17) is obtained by the formula $T_{m+1}^{1}(j+1) = D_{out}(j+1) + D_{m+1}^{42}(j+2)$ with $k=m+1$ in expression (5.19). Note that, in general, expressions (5.1), (5.2), (5.3), (5.17), (5.18) and (5.19) are a function of both k and j. Therefore, the delay of any combination of k and j that would make any term in each element infeasible or undefined will not include that corresponding element in its representation. For example, we are interested in finding $D_2^{42}(3)$ for a three–stage micropipeline using expression (5.19), i.e., $m=3$, $k=2$ and $j=1$. The expression will be

$$D_2^{42}(3) = Max(0, min(DB_1(2) - DF_2(2) + D_1^{42}(3),$$

$$DB_1(2) - [B_2(2) - B_2(1)] - DF_3(1) + \sum_{l=1}^{2} D_l^{42}(4-l)))$$

The expression $DB_1(2) - \sum_{l=2}^{3}[B_l(4-l) - B_l(3-l)] - DF_4(0) + \sum_{l=1}^{3} D_l^{42}(4-l)$ should not be included since $B_3(0)$, $DF_4(0)$ and $D_3^{42}(1)$ are not defined.

5.1.2 Several Approximations To $D_k^{42}(j+2)$

In the early design phase, an engineer may be more interested in quickly obtaining an approximation of the pipeline's performance, rather than in finding its exact

bounds. Exact performance bounds can be obtained during the final design phase by using simulation tool.

Since $T^{1}_{m+1}(j+1) = D_{out}(j+1) + D^{42}_{m+1}(j+2)$, the upper and lower bounds of output loop delay $T^{1}_{m+1}(j+1)$ become known once the upper and lower bounds of $D^{42}_{m+1}(j+2)$ are known. A similar argument can be applied to equivalent input delays for each stage $D^{in}_{k}(j+2) = B_k(j+1) + D^{42}_{k}(j+2) + F_k(j+2)$. Therefore, we only need to focus on two representations ((5.3) and (5.19)) of logic delay $D^{42}_{k}(j+2)$, $1 \le k \le m+1$. It should be noted that the exact values of $D^{42}_{k}(j+2)$ obtained from expressions (5.3) and (5.19) should be the same for any given k and j, even though their representations are different. The major difference is that the sign in front of the logic delay term in expression (5.3) is negative and in expression (5.19) it is positive. Since all of the logic delays are defined as being greater than or equal to zero, the sign difference allows us to approximate the upper and lower bounds of $D^{42}_{k}(j+2)$ from right–hand and left–hand sides. Several approximations are discussed as follows. Assuming that the upper and lower bounds of each stage are given, for example,

$$D^{min}_{in} \le D_{in}(j+2) \le D^{Max}_{in},$$

$$D^{min}_{out} \le D_{out}(j+1) \le D^{Max}_{out},$$

$$B^{min}_{i} \le B_i(j+1) \le B^{Max}_{i}, \text{ and}$$

$$F^{min}_{i} \le F_i(j+1) \le F^{Max}_{i},\ 1 \le i \le m.$$

Several notations are adopted.

$Max(f(\cdot))|_j$: The exact upper bound of $f(\cdot)$ for all js; sometimes, also written as f^{Max} for convenience.

$min(f(\cdot))|_j$: The exact lower bound of $f(\cdot)$ for all js; sometimes, also written as f^{min} for convenience.

$UP_x(f(\cdot))|_j$: An expression approximating $Max(f(\cdot))|_j$; based on approximation x.

$LW_x(f(\cdot))|_j$: An expression approximating $min(f(\cdot))|_j$; based on approximation x.

Note that $UP_x(f(\cdot))|_j$ could be greater than, equal to or less than $Max(f(\cdot))|_j$ depending on approximation x. A similar relationship can be applied to $LW_x(f(\cdot))|_j$ and $min(f(\cdot))|_j$.

<Approximation 1>:

From expression (5.3), if the logic delay term is dropped, we have

$$
\begin{aligned}
& D_k^{42}(j+2) \\
& \le Max(0, \\
& \quad DB_{k-1}(j+1) - DF_k(j+1), \\
& \quad DB_{k-2}(j+1) + \sum_{l=k-1}^{k-1} (F_l(j+2) - F_l(j+1)) - DF_k(j+1), \\
& \quad \cdots\cdots \\
& \quad DB_1(j+1) + \sum_{l=2}^{k-1} (F_l(j+2) - F_l(j+1)) - DF_k(j+1), \\
& \quad DB_0(j+1) + \sum_{l=1}^{k-1} (F_l(j+2) - F_l(j+1)) - DF_k(j+1)) \qquad (5.20)
\end{aligned}
$$

Apparently, the largest value of $D_k^{42}(j+2)$ for all js, denoted as $Max(D_k^{42}(j+2))|_j$, will be less than or equal to the largest value of the right–hand–side of expression (5.20). Also, the smallest value of $D_k^{42}(j+2)$ for all js, denoted as $min(D_k^{42}(j+2))|_j$, will be less than or equal to the smallest value of right–hand–side of expression (5.20). Since we assume that all stage–delays for each stage are independent (e.g., $F_p(j+1)$ and $F_q(j+1)$ are independent where $p \neq q$, $1 \le p \le m$ and $1 \le q \le m$. The same assumption can be applied to $B_p(j+1)$ and $B_q(j+1)$, $F_p(j+1)$ and $B_p(j+1)$, and

$F_p(j+1)$ and $B_q(j+1)$), we can make the positive term maximum and the negative term minimum to approximate maximum value. That is,

$$
\begin{aligned}
& Max(D_k^{42}(j+2))|_j \\
& \leq Max(0, \\
& \qquad DB_{k-1}^{Max} - DF_k^{min}, \\
& \qquad DB_{k-2}^{Max} + \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - DF_k^{min}, \\
& \qquad \cdots\cdots \\
& \qquad DB_1^{Max} + \sum_{l=2}^{k-1} (F_l^{Max} - F_l^{min}) - DF_k^{min}, \\
& \qquad DB_0^{Max} + \sum_{l=1}^{k-1} (F_l^{Max} - F_l^{min}) - DF_k^{min}) \\
& = UP_1(D_k^{42}(j+2))|_j
\end{aligned}
\tag{5.21}
$$

The $UP_1(D_k^{42}(j+2))|_j$ is the representation of *Max(0,.....)* in (5.21) for *approximation 1*, i.e., $Max(D_k^{42}(j+2))|_j \leq UP_1(D_k^{42}(j+2))|_j$.

If we make the positive term minimum and the negative term maximum, then we reduce the approximation to its minimum value. That is,

$$
\begin{aligned}
& min(D_k^{42}(j+2))|_j \\
& \leq Max(0, \\
& \qquad DB_{k-1}^{min} - DF_k^{Max}, \\
& \qquad DB_{k-2}^{min} - \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - DF_k^{Max}, \\
& \qquad \cdots\cdots \\
& \qquad DB_1^{min} - \sum_{l=2}^{k-1} (F_l^{Max} - F_l^{min}) - DF_k^{Max},
\end{aligned}
$$

$$DB_0^{min} - \sum_{l=1}^{k-1}(F_l^{Max} - F_l^{min}) - DF_k^{Max})$$

$$= LW_1(D_k^{42}(j+2))|_j \tag{5.22}$$

The $LW_1(D_k^{42}(j+2))|_j$ is the representation of *Max(0,.....)* in (5.22) for *approximation 1,* i.e., $min(D_k^{42}(j+2))|_j \le LW_1(D_k^{42}(j+2))|_j$.

where $DB_0^{min} = D_{in}^{min}$,

$DB_0^{Max} = D_{in}^{Max}$,

$DF_{m+1}^{min} = D_{out}^{min}$,

$DF_{m+1}^{Max} = D_{out}^{Max}$,

$DB_i^{Max} = F_i^{Max} + B_i^{Max} = DF_i^{Max}$,

$DB_i^{min} = F_i^{min} + B_i^{min} = DF_i^{min}$, $1 \le i \le m$.

<Approximation 2>:

From expression (5.3), if the logic delay term is dropped, except for $D_{k+1}^{24}(j+1)$, and since $D_k^{out}(j+1) = DF_k(j+1) + D_{k+1}^{24}(j+1)$, we have a more accurate expression for $D_k^{42}(j+2)$. That is,

$D_k^{42}(j+2)$

$\le Max(0,$

$DB_{k-1}(j+1) - D_k^{out}(j+1)$,

$DB_{k-2}(j+1) + \sum_{l=k-1}^{k-1}(F_l(j+2) - F_l(j+1)) - D_k^{out}(j+1)$,

. . . . . .

$DB_1(j+1) + \sum_{l=2}^{k-1}(F_l(j+2) - F_l(j+1)) - D_k^{out}(j+1)$,

$DB_0(j+1) + \sum_{l=1}^{k-1}(F_l(j+2) - F_l(j+1)) - D_k^{out}(j+1))$ (5.23)

$$
\begin{aligned}
& Max(D_k^{42}(j+2))|_j \\
& \le Max(0, \\
& \qquad DB_{k-1}^{Max} - min(D_k^{out}(j+1))|_j, \\
& \qquad DB_{k-2}^{Max} + \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - min(D_k^{out}(j+1))|_j, \\
& \qquad \cdots\cdots \\
& \qquad DB_1^{Max} + \sum_{l=2}^{k-1} (F_l^{Max} - F_l^{min}) - min(D_k^{out}(j+1))|_j, \\
& \qquad DB_0^{Max} + \sum_{l=1}^{k-1} (F_l^{Max} - F_l^{min}) - min(D_k^{out}(j+1))|_j) \\
& = UP_2(D_k^{42}(j+2))|_j
\end{aligned} \tag{5.24}
$$

$$
\begin{aligned}
& min(D_k^{42}(j+2))|_j \\
& \le Max(0, \\
& \qquad DB_{k-1}^{min} - Max(D_k^{out}(j+1))|_j, \\
& \qquad DB_{k-2}^{min} - \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - Max(D_k^{out}(j+1))|_j, \\
& \qquad \cdots\cdots \\
& \qquad DB_1^{min} - \sum_{l=2}^{k-1} (F_l^{Max} - F_l^{min}) - Max(D_k^{out}(j+1))|_j, \\
& \qquad DB_0^{min} - \sum_{l=1}^{k-1} (F_l^{Max} - F_l^{min}) - Max(D_k^{out}(j+1))|_j) \\
& = LW_2(D_k^{42}(j+2))|_j
\end{aligned} \tag{5.25}
$$

where $min(D_k^{out}(j+1))|_j = F_k^{min} + B_k^{min} + min(D_{k+1}^{24}(j+1))|_j$,

$Max(D_k^{out}(j+1))|_j = F_k^{Max} + B_k^{Max} + min(D_{k+1}^{24}(j+1))|_j$,

$min(D_{m+1}^{out}(j+1))|_j = D_{out}^{min}$,

$$Max(D^{out}_{m+1}(j+1))|_j = D^{Max}_{out}.$$

To utilize this approximation, the minimum value of $D^{24}_k(j+1)$ should be known in advance. Since we are interested in a micropipeline which is initially reset, the minimum value of $D^{24}_k(j+1)$ is zero (e.g., $D^{24}_k(1) = 0$). That is, the equal sign of expression (5.21) holds for initially reset micropipelines. Note that the assumption of $min(D^{out}_k(j+1))|_j = F^{min}_k + B^{min}_k + min(D^{24}_{k+1}(j+1))|_j$ and $Max(D^{out}_k(j+1))|_j = F^{Max}_k + B^{Max}_k + min(D^{24}_{k+1}(j+1))|_j$ are for the convenience of calculation. The exact expression will be discussed in a later section.

<Approximation 3>:

From expression (5.19), if the logic delay term is dropped, we have

$$\begin{aligned}
&D^{42}_k(j+2) \\
&\geq Max(0,\ min(\\
&\quad DB_{k-1}(j+1) - DF_k(j+1), \\
&\quad DB_{k-1}(j+1) - \sum_{l=k}^{k}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_{k+1}(j), \\
&\quad \cdots\cdots \\
&\quad DB_{k-1}(j+1) - \sum_{l=k}^{m-1}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_m(j+1-m+k), \\
&\quad DB_{k-1}(j+1) - \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_{m+1}(j-m+k)))
\end{aligned} \tag{5.26}$$

$$\begin{aligned}
&Max(D^{42}_k(j+2))|_j \\
&\geq Max(0,\ min(DB^{Max}_{k-1} - DF^{min}_k, \\
&\qquad DB^{Max}_{k-1} - \sum_{l=k}^{k}(B^{min}_l - B^{Max}_l) - DF^{min}_{k+1}, \\
&\qquad \cdots\cdots
\end{aligned}$$

$$DB_{k-1}^{Max} - \sum_{l=k}^{m-1}(B_l^{min} - B_l^{Max}) - DF_m^{min},$$

$$DB_{k-1}^{Max} - \sum_{l=k}^{m}(B_l^{min} - B_l^{Max}) - DF_{m+1}^{min}))$$

$$= UP_3(D_k^{42}(j+2))|_j \tag{5.27}$$

$$min(D_k^{42}(j+2))|_j$$

$$\geq Max(0,\ min(DB_{k-1}^{min} - DF_k^{Max},$$

$$DB_{k-1}^{min} - \sum_{l=k}^{k}(B_l^{Max} - B_l^{min}) - DF_{k+1}^{Max},$$

$$\cdots\cdots$$

$$DB_{k-1}^{min} - \sum_{l=k}^{m-1}(B_l^{Max} - B_l^{min}) - DF_m^{Max},$$

$$DB_{k-1}^{min} - \sum_{l=k}^{m}(B_l^{Max} - B_l^{min}) - DF_{m+1}^{Max}))$$

$$= LW_3(D_k^{42}(j+2))|_j \tag{5.28}$$

<Approximation 4>:

From expression (5.19), if the logic delay term is dropped, except for $D_{k-1}^{42}(j+2)$, and since $D_{k-1}^{in}(j+2) = DB_{k-1}(j+1) + D_{k-1}^{42}(j+2)$, we have a more accurate expression for $D_k^{42}(j+2)$. That is,

$D_k^{42}(j+2)$

$\geq Max(0,\ min($

$$D_{k-1}^{in}(j+2) - DF_k(j+1),$$

$$D_{k-1}^{in}(j+2) - \sum_{l=k}^{k}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_{k+1}(j),$$

$\cdots\cdots$

$$D^{in}_{k-1}(j+2) - \sum_{l=k}^{m-1}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_m(j+1-m+k),$$

$$D^{in}_{k-1}(j+2) - \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DF_{m+1}(j-m+k)))$$

(5.29)

$$Max(D^{42}_k(j+2))|_j$$

$$\geq Max(0,\ min(Max(D^{in}_{k-1}(j+2))|_j - DF^{min}_k,$$

$$Max(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{k}(B^{min}_l - B^{Max}_l) - DF^{min}_{k+1},$$

$$\cdots\cdots$$

$$Max(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{m-1}(B^{min}_l - B^{Max}_l) - DF^{min}_m,$$

$$Max(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{m}(B^{min}_l - B^{Max}_l) - DF^{min}_{m+1}))$$

$$= UP_4(D^{42}_k(j+2))|_j \tag{5.30}$$

$$min(D^{42}_k(j+2))|_j$$

$$\geq Max(0,\ min(min(D^{in}_{k-1}(j+2))|_j - DF^{Max}_k,$$

$$min(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{k}(B^{Max}_l - B^{min}_l) - DF^{Max}_{k+1},$$

$$\cdots\cdots$$

$$min(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{m-1}(B^{Max}_l - B^{min}_l) - DF^{Max}_m,$$

$$min(D^{in}_{k-1}(j+2))|_j - \sum_{l=k}^{m}(B^{Max}_l - B^{min}_l) - DF^{Max}_{m+1}))$$

$$= LW_4(D^{42}_k(j+2))|_j \tag{5.31}$$

where $D_0^{in,min} = D_{in}^{min}$,

$D_0^{in,Max} = D_{in}^{Max}$,

$min(D_i^{in}(j+2))|_j = DB_i^{min} + min(D_i^{42}(j+2))|_j$,

$Max(D_i^{in}(j+2))|_j = DB_i^{Max} + min(D_i^{42}(j+2))|_j$, $1 \le i \le m$.

To utilize this approximation, the minimum value of $D_{k-1}^{42}(j+2)$ should be known in advance. Since, in general, $D_{k-1}^{42}(j+2) \neq 0$ given $j=q$ and $k=w$ where $q \ge 0$ and $1 \le w \le m+1$, the equal signs from expressions (5.26) to (5.31) may not hold for initially reset micropipelines. The assumption of $min(D_i^{in}(j+2))|_j = DB_i^{min} + min(D_i^{42}(j+2))|_j$ and $Max(D_i^{in}(j+2))|_j = DB_i^{Max} + min(D_i^{42}(j+2))|_j$ is for the convenience of calculation only. Exact expressions will be discussed in a later section.

Besides the four approximations given above, other means can make the approximation more accurate, with the price of also making it more complex. For example, instead of ignoring the whole logic delay term in expression (5.19), we can take into account more logic delays when making the approximation.

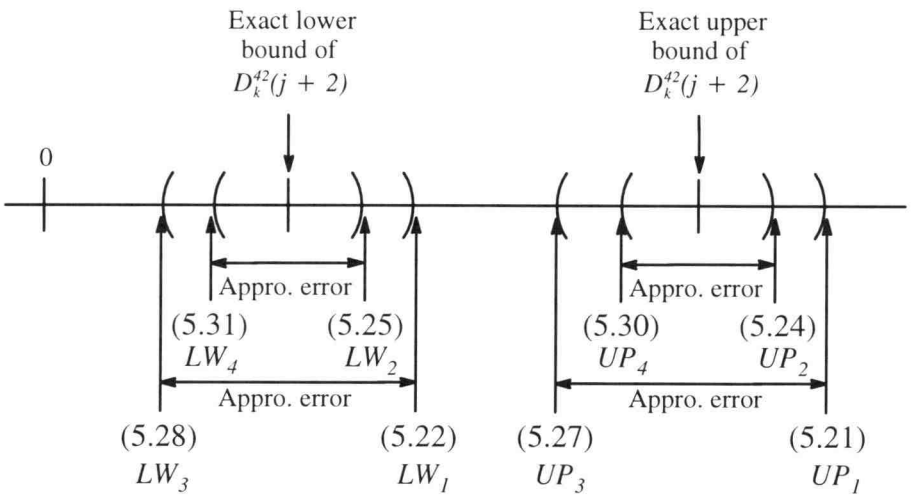
The above approximations are summarized in Figure 5.1. From the above discussion, we know that there are four approximations of the exact upper bound and four of the exact lower bound. If the micropipeline is considered to be reset initially, then $Max(D_k^{42}(j+2))|_j = UP_1(D_k^{42}(j+2))|_j = UP_2(D_k^{42}(j+2))|_j$. Otherwise, the errors based on these four approximations will be shown as follows. Note that the error itself is also an approximation.

Upper Bound: Best Approximation Error(x100%): [(5.24) – (5.30)]/(5.30)

Upper Bound: Worst Approximation Error(x100%): [(5.21) – (5.27)]/(5.27)

Lower Bound: Best Approximation Error(x100%): [(5.25) – (5.31)]/(5.31)

Lower Bound: Worst Approximation Error(x100%): [(5.22) – (5.28)]/(5.28)

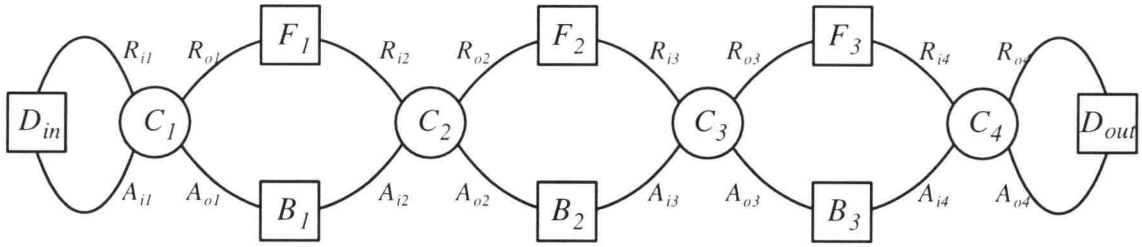


* If a micropipeline is initially reset, expressions (5.21) and (5.24) will coincide with exact upper bound.

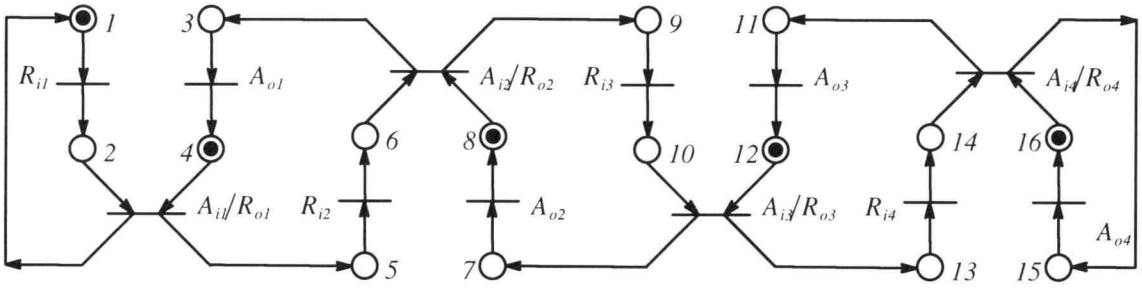
Figure 5.1: Relative relationship among different approximations of $D_k^{42}(j+2)$.

5.1.3 Several Approximations To $T^1_{m+1}(j+1)$

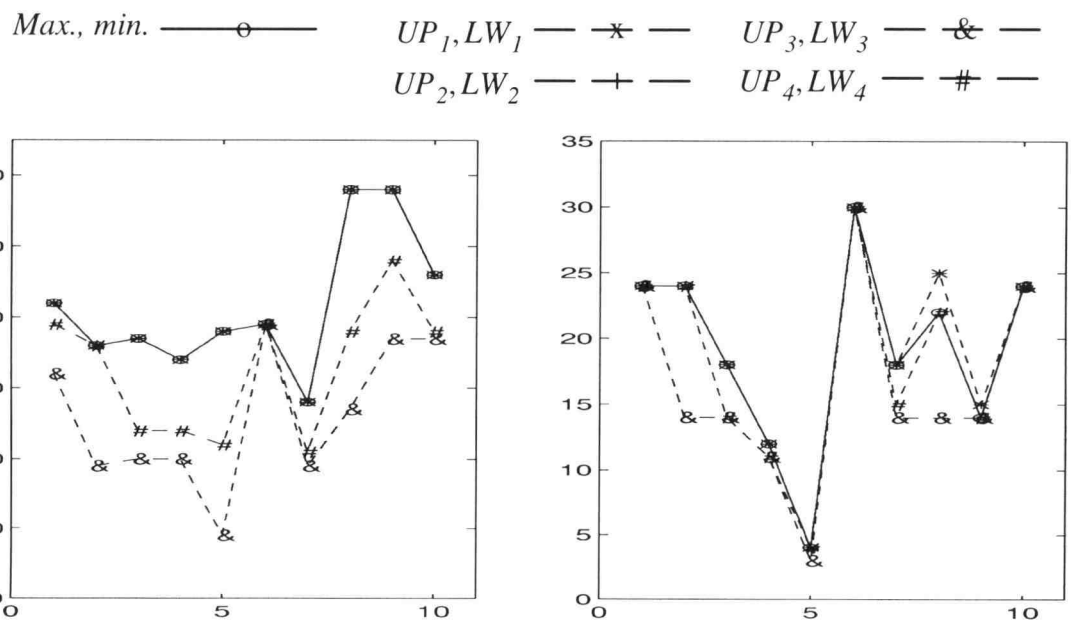
As mentioned before, a method for finding the exact bounds of a general system with choice modeled in Petri net has been developed [21, 22]. However, the algorithm used in this work cannot guarantee the exact bounds for a system which is data dependent and contains the mechanism of mutual exclusion. For such systems, only approximate results can be determined. A tool called "CTSE" accepts the input of a Petri net model and implements this algorithm [21, 22]. According to the author, CTSE is not an acronym. However, the "C" stands for "conditional" or "choice" and "TSE" stands for "time separation of events". We would like to compare our approximations, based on Figure 5.1 and the fact of $T^1_{m+1}(j+1) = D_{out}(j+1) + D^{42}_{m+1}(j+2)$, with the exact values obtained through CTSE. Figure 5.2(a) shows a three–stage linear pipeline ($m=3$) used for comparison. The corresponding Petri net model for this pipeline is depicted in Figure 5.2(b). The token marking in Figure 5.2(b) indicates the initial condition of the pipeline. Table 5.1 shows ten simulations corresponding to different stage–delay combinations. The first number in brackets represents the minimum value and the second number represents the maximum value. The result is also plotted in Figure 5.2(c). Due to the initial reset condition, the output loop delay using $UP_1(\cdot)$ approach has the same value as the exact maximum bound, as mentioned earlier. Besides, since $D^{24}_5(j+1) = 0$ by default in the case of $m=3$, the two approaches $UP_1(\cdot)$ and $UP_2(\cdot)$ are the same. This is also true for $LW_1(\cdot)$ and $LW_2(\cdot)$. By observing the second simulation case in which $UP_3(\cdot) = 19 < 24 = LW_{1,2}(\cdot)$, it is possible that $UP_{3,4}(\cdot) \leq LW_{1,2}(\cdot)$. Note that it is not necessary true that $LW_1(\cdot)$ has the same value as the exact minimum bound (refer to the 8th and 9th simulation results); although it does in most cases in our simulations. These simulations also show that $UP_4(\cdot)$ approaches the exact maximum bound better than $UP_3(\cdot)$ and $LW_4(\cdot)$ approaches the exact minimum bound better than $LW_3(\cdot)$.



(a)



(b)



$Max., UP_1, UP_2, UP_3$ and UP_4 of $T^l_{m+1}(j+1)$

$min., LW_1, LW_2, LW_3$ and LW_4 of $T^l_{m+1}(j+1)$

(c)

Figure 5.2: Comparison of our approximations with the exact bounds.
(a) A 3–stage linear pipeline.
(b) Petri net model.
(c) Result comparison.

Table 5.1: Comparison of our approximations with the exact bounds using a three–stage linear pipeline as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_{in}	[2 4]	[2 4]	[2 4]	[22 24]	[6 7]	[6 7]	[6 7]	[36 47]	[36 37]	[6 7]
F_1	[12 24]	[20 24]	[22 24]	[12 14]	[10 20]	[10 20]	[10 11]	[20 22]	[20 22]	[20 22]
B_1	[4 5]	[4 5]	[4 5]	[2 5]	[2 5]	[2 5]	[10 15]	[4 15]	[4 15]	[4 15]
F_2	[22 29]	[25 29]	[12 16]	[12 16]	[2 14]	[12 14]	[12 13]	[22 30]	[22 30]	[22 30]
B_2	[2 4]	[2 4]	[2 4]	[2 4]	[3 7]	[3 7]	[4 7]	[1 7]	[1 7]	[1 7]
F_3	[20 26]	[10 13]	[10 14]	[10 14]	[2 3]	[29 33]	[12 13]	[12 13]	[12 23]	[22 23]
B_3	[4 6]	[4 6]	[4 6]	[1 6]	[1 6]	[1 6]	[2 6]	[2 14]	[2 14]	2 14]
D_{out}	[2 4]	[2 4]	[2 4]	[2 4]	[2 4]	[2 4]	[2 4]	[2 4]	[2 4]	[12 14]
Max	42	36	37	34	38	39	28	58	58	46
UP_1	42	36	37	34	38	39	28	58	58	46
UP_2	42	36	37	34	38	39	28	58	58	46
UP_3	32	19	20	20	9	39	19	27	37	37
UP_4	39	36	24	24	22	39	21	38	48	38
min	24	24	18	12	4	30	18	22	14	24
LW_1	24	24	18	12	4	30	18	25	15	24
LW_2	24	24	18	12	4	30	18	25	15	24
LW_3	24	14	14	11	3	30	14	14	14	24
LW_4	24	24	14	11	4	30	15	22	14	24

To simplify the discussion of design procedure stated in a later section, only $UP_1(D^{42}_{m+1}(j+2))|_j$ is considered from now on, unless specified otherwise. We also assume that the minimum value of $D^{42}_{m+1}(j+2)$ is zero. As a result, since $T^1_{m+1}(j+1) = D_{out}(j+1) + D^{42}_{m+1}(j+2)$, output loop delay has a lower bound,

$$T^1_{m+1}(j+1) \geq D_{out}(j+1) \tag{5.32}$$

and an upper bound based on (5.20) for $k = m+1$,

$$T^{l}_{m+1}(j+1) \le Max(D_{out}(j+1),$$

$$DB_m(j+1),$$

$$DB_{m-1}(j+1) + \sum_{l=m}^{m}(F_l(j+2) - F_l(j+1)),$$

$$DB_{m-2}(j+1) + \sum_{l=m-1}^{m}(F_l(j+2) - F_l(j+1)),$$

$$\cdots\cdots$$

$$DB_1(j+1) + \sum_{l=2}^{m}(F_l(j+2) - F_l(j+1)),$$

$$D_{in}(j+2) + \sum_{l=1}^{m}(F_l(j+2) - F_l(j+1))) \qquad (5.33)$$

Expressions (5.32) and (5.33) allow us to find upper and lower bounds of *individual* output loop delay for each j if both the forward delay $F_i(j+1)$ and backward delay $B_i(j+1)$ for each j are known.

If each stage's upper and lower bounds are given, then expression (5.32) and (5.33) can be rewritten as

$$T^{l}_{m+1}(j+1) \ge D^{min}_{out} \qquad (5.34)$$

$$T^{l}_{m+1}(j+1) \le Max(D^{Max}_{out},$$

$$DB^{Max}_{m},$$

$$DB^{Max}_{m-1} + \sum_{l=m}^{m}(F^{Max}_{l} - F^{min}_{l}),$$

$$DB^{Max}_{m-2} + \sum_{l=m-1}^{m}(F^{Max}_{l} - F^{min}_{l}),$$

$$\cdots\cdots$$

$$DB^{Max}_{1} + \sum_{l=2}^{m}(F^{Max}_{l} - F^{min}_{l}),$$

$$D_{in}^{Max} + \sum_{l=1}^{m}(F_l^{Max} - F_l^{min})) \tag{5.35}$$

Expressions (5.34) and (5.35) allow us to find the upper and lower bounds of *overall* output loop delay if the upper and lower bounds of both forward delay $F_i(j + 1)$ and backward delay $B_i(j + 1)$ are known. That is, each output loop delay corresponding to a different j must be bounded by expressions (5.34) and (5.35). Figure 5.3 shows the relative relationship among the expressions (5.32), (5.33), (5.34) and (5.35).

Note that expression (5.34) is a sufficient (not necessary) condition for a micropipeline to satisfy the required lower bound. This is obvious since expression (5.34) drops the logic delay term $D_{m+1}^{42}(j + 2)$. In other words, $D_{m+1}^{42}(j + 2)$ is assumed to have a lower bound zero. The upper bound of output loop delay, shown in expression (5.35) (note that the "=" sign holds for initial reset pipeline), however, is a sufficient and necessary condition for a micropipeline to satisfy the required upper bound. The "=" sign in expression (5.35) can be justified if we substitute $F_i(1) = F_i^{min}$, $F_i(2) = F_i^{Max}$, $B_i(1) = B_i^{Max}$, $DB_0(1) = D_{in}^{Max}$ and $DF_{m+1}(1) = D_{out}^{Max}$ into expression (5.1) for $j=0$, where $1 \le i \le m$. Note that $D_k^{24}(1) = 0$, $1 \le k \le m + 1$.

5.1.4 Several Approximations To $D_k^{24}(j + 2)$

Just like the logic delay $D_k^{42}(j + 2)$, there are four approximations to $D_k^{24}(j + 2)$. From expression (5.2), we have

$D_k^{24}(j + 2)$

$= Max(0,$

$$DF_k(j + 1) - DB_{k-1}(j + 1) - \sum_{l=k-1}^{k-1} D_l^{42}(j + 1 + k - l),$$

$$DF_{k+1}(j) + \sum_{l=k}^{k}(B_l(j + 1 + k - l) - B_l(j + k - l)) - DB_{k-1}(j + 1)$$

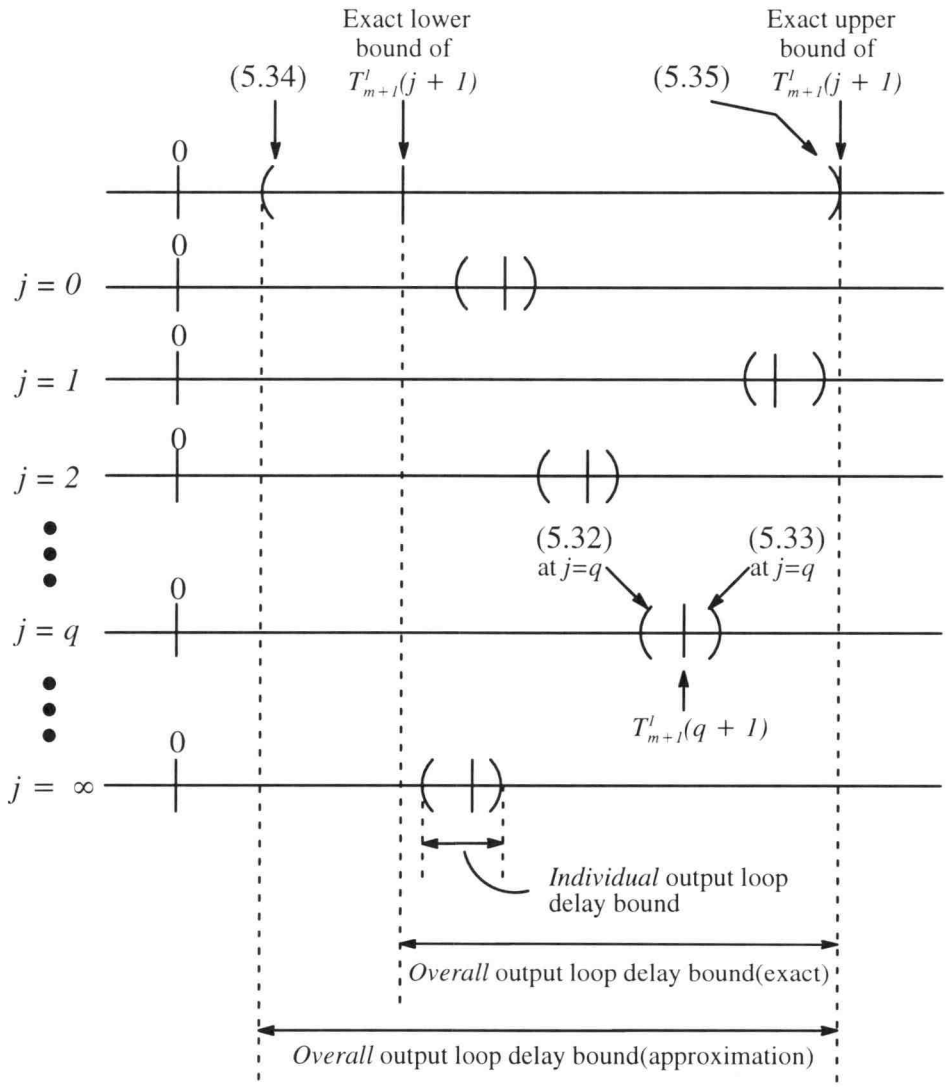


Figure 5.3: The relationship between individual and overall output loop delay bounds.

$$- \sum_{l=k-1}^{k} D_l^{42}(j+1+k-l),$$

$$\cdots\cdots$$

$$DF_m(j+1-m+k) + \sum_{l=k}^{m-1}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m-1} D_l^{42}(j+1+k-l),$$

$$DF_{m+1}(j-m+k) + \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m} D_l^{42}(j+1+k-l))$$

<Approximation 1>:

If the logic delay term is dropped, we then have

$$Max(D_k^{24}(j+2))|_j$$

$$\leq Max(0,$$

$$DF_k^{Max} - DB_{k-1}^{min},$$

$$DF_{k+1}^{Max} + \sum_{l=k}^{k}(B_l^{Max} - B_l^{min}) - DB_{k-1}^{min},$$

$$\cdots\cdots$$

$$DF_m^{Max} + \sum_{l=k}^{m-1}(B_l^{Max} - B_l^{min}) - DB_{k-1}^{min},$$

$$DF_{m+1}^{Max} + \sum_{l=k}^{m}(B_l^{Max} - B_l^{min}) - DB_{k-1}^{min})$$

$$= UP_1(D_k^{24}(j+2))|_j \tag{5.36}$$

$$min(D_k^{24}(j+2))|_j$$

$$\leq Max(0,$$

$$DF_k^{min} - DB_{k-1}^{Max},$$

$$DF_{k+1}^{min} + \sum_{l=k}^{k} (B_l^{min} - B_l^{Max}) - DB_{k-1}^{Max},$$

. . . . . .

$$DF_{m}^{min} + \sum_{l=k}^{m-1} (B_l^{min} - B_l^{Max}) - DB_{k-1}^{Max},$$

$$DF_{m+1}^{min} + \sum_{l=k}^{m} (B_l^{min} - B_l^{Max}) - DB_{k-1}^{Max})$$

$$= LW_1(D_k^{24}(j+2))|_j \tag{5.37}$$

where $1 \le k \le m+1$

$$DB_0^{min} = D_{in}^{min},$$

$$DB_0^{Max} = D_{in}^{Max},$$

$$DF_{m+1}^{min} = D_{out}^{min},$$

$$DF_{m+1}^{Max} = D_{out}^{Max}.$$

<Approximation 2>:

If the logic delay term is dropped except for the term $D_{k-1}^{42}(j+2)$ and since $D_{k-1}^{in}(j+2) = DB_{k-1}(j+1) + D_{k-1}^{42}(j+2)$, we have a more accurate expression for $D_k^{24}(j+2)$.

$Max(D_k^{24}(j+2))|_j$

$\le Max(0,$

$$DF_k^{Max} - min(D_{k-1}^{in}(j+2))|_j,$$

$$DF_{k+1}^{Max} + \sum_{l=k}^{k} (B_l^{Max} - B_l^{min}) - min(D_{k-1}^{in}(j+2))|_j,$$

. . . . . .

$$DF_{m}^{Max} + \sum_{l=k}^{m-1} (B_l^{Max} - B_l^{min}) - min(D_{k-1}^{in}(j+2))|_j,$$

$$DF^{Max}_{m+1} + \sum_{l=k}^{m}(B^{Max}_{l} - B^{min}_{l}) - min(D^{in}_{k-1}(j+2))|_j)$$

$$= UP_2(D^{24}_k(j+2))|_j \tag{5.38}$$

$$min(D^{24}_k(j+2))|_j$$

$$\leq Max(0,$$

$$DF^{min}_{k} - Max(D^{in}_{k-1}(j+2))|_j,$$

$$DF^{min}_{k+1} + \sum_{l=k}^{k}(B^{min}_{l} - B^{Max}_{l}) - Max(D^{in}_{k-1}(j+2))|_j,$$

$$\cdots\cdots$$

$$DF^{min}_{m} + \sum_{l=k}^{m-1}(B^{min}_{l} - B^{Max}_{l}) - Max(D^{in}_{k-1}(j+2))|_j,$$

$$DF^{min}_{m+1} + \sum_{l=k}^{m}(B^{min}_{l} - B^{Max}_{l}) - Max(D^{in}_{k-1}(j+2))|_j)$$

$$= LW_2(D^{24}_k(j+2))|_j \tag{5.39}$$

where $min(D^{in}_k(j+2))|_j = F^{min}_k + B^{min}_k + min(D^{42}_k(j+2))|_j$,

$Max(D^{in}_k(j+2))|_j = F^{Max}_k + B^{Max}_k + min(D^{42}_k(j+2))|_j$,

$min(D^{in}_0(j+1))|_j = D^{min}_{in}$,

$Max(D^{in}_0(j+1))|_j = D^{Max}_{in}$.

To utilize this approximation, the minimum value of $D^{42}_k(j+2)$ should be known in advance. Note that the assumption of $min(D^{in}_k(j+2))|_j = F^{min}_k + B^{min}_k + min(D^{42}_k(j+2))|_j$ and $Max(D^{in}_k(j+2))|_j = F^{Max}_k + B^{Max}_k + min(D^{42}_k(j+2))|_j$ are for the convenience of calculation. The exact expression will be discussed in a later section.

From expression (5.18), the other representation of $D^{24}_k(j+2)$ is shown below.

$D^{24}_k(j+2)$

$= Max(0, min($

$$DF_k(j+1) - DB_{k-1}(j+1) + \sum_{l=k+1}^{k+1} D_l^{24}(j+1),$$

$$DF_k(j+1) - \sum_{l=k-1}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_{k-2}(j+1) + \sum_{l=k}^{k+1} D_l^{24}(j+1),$$

$$\cdots\cdots$$

$$DF_k(j+1) - \sum_{l=2}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_1(j+1) + \sum_{l=3}^{k+1} D_l^{24}(j+1),$$

$$DF_k(j+1) - \sum_{l=1}^{k-1} (F_l(j+2) - F_l(j+1)) - DB_0(j+1) + \sum_{l=2}^{k+1} D_l^{24}(j+1)))$$

<Approximation 3>:

If the logic delay term is dropped, we then have

$$\begin{aligned}
&Max(D_k^{24}(j+2))|_j \\
&\geq Max(0, min(DF_k^{Max} - DB_{k-1}^{min}, \\
&\qquad DF_k^{Max} + \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - DB_{k-2}^{min}, \\
&\qquad \cdots\cdots \\
&\qquad DF_k^{Max} + \sum_{l=2}^{k-1} (F_l^{Max} - F_l^{min}) - DB_1^{min}, \\
&\qquad DF_k^{Max} + \sum_{l=1}^{k-1} (F_l^{Max} - F_l^{min}) - DB_0^{min})) \\
&= UP_3(D_k^{24}(j+2))|_j \qquad (5.40) \\
&min(D_k^{24}(j+2))|_j \\
&\geq Max(0, min(DF_k^{min} - DB_{k-1}^{Max}, \\
&\qquad DF_k^{min} - \sum_{l=k-1}^{k-1} (F_l^{Max} - F_l^{min}) - DB_{k-2}^{Max}, \\
&\qquad \cdots\cdots
\end{aligned}$$

$$DF_k^{min} - \sum_{l=2}^{k-1}(F_l^{Max} - F_l^{min}) - DB_1^{Max},$$

$$DF_k^{min} - \sum_{l=1}^{k-1}(F_l^{Max} - F_l^{min}) - DB_0^{Max}))$$

$$= LW_3(D_k^{24}(j+2))|_j \tag{5.41}$$

<Approximation 4>:

If the logic delay term is dropped except for the term $D_{k+1}^{24}(j+1)$ and since $D_k^{out}(j+1) = DF_k(j+1) + D_{k+1}^{24}(j+1)$, we have a more accurate expression for $D_k^{24}(j+2)$.

$Max(D_k^{24}(j+2))|_j$

$\geq Max(0, min(Max(D_k^{out}(j+1))|_j - DB_{k-1}^{min},$

$$Max(D_k^{out}(j+1))|_j + \sum_{l=k-1}^{k-1}(F_l^{Max} - F_l^{min}) - DB_{k-2}^{min},$$

$$\cdots\cdots$$

$$Max(D_k^{out}(j+1))|_j + \sum_{l=2}^{k-1}(F_l^{Max} - F_l^{min}) - DB_1^{min},$$

$$Max(D_k^{out}(j+1))|_j + \sum_{l=1}^{k-1}(F_l^{Max} - F_l^{min}) - DB_0^{min}))$$

$$= UP_4(D_k^{24}(j+2))|_j \tag{5.42}$$

$min(D_k^{24}(j+2))|_j$

$\geq Max(0, min(min(D_k^{out}(j+1))|_j - DB_{k-1}^{Max},$

$$min(D_k^{out}(j+1))|_j - \sum_{l=k-1}^{k-1}(F_l^{Max} - F_l^{min}) - DB_{k-2}^{Max},$$

$$\cdots\cdots$$

$$min(D_k^{out}(j+1))|_j - \sum_{l=2}^{k-1}(F_l^{Max} - F_l^{min}) - DB_1^{Max},$$

$$min(D_k^{out}(j+1))|_j - \sum_{l=1}^{k-1}(F_l^{Max} - F_l^{min}) - DB_0^{Max}))$$

$$= LW_4(D_k^{24}(j+2))|_j \tag{5.43}$$

where $D_{m+1}^{out,min} = D_{out}^{min}$,

$D_{m+1}^{out,Max} = D_{out}^{Max}$,

$min(D_i^{out}(j+1))|_j = DB_i^{min} + min(D_{i+1}^{24}(j+1))|_j$,

$Max(D_i^{out}(j+1))|_j = DB_i^{Max} + min(D_{i+1}^{24}(j+1))|_j$, $1 \le i \le m$.

To utilize this approximation, the minimum value of $D_{i+1}^{24}(j+1)$ should be known in advance. Since $D_1^{24}(1) = 0$, the equal sign of expressions (5.41) to (5.43) hold for initially reset micropipelines. That is, if a pipeline is initially empty, then $min(D_k^{24}(j+2))|_j = LW_3(D_k^{24}(j+2))|_j = LW_4(D_k^{24}(j+2))|_j$. Figure 5.4 demonstrates the relationship between exact bounds of $D_k^{24}(j+2)$ and expressions (5.36) to (5.43). The assumption of $min(D_i^{out}(j+1))|_j = DB_i^{min} + min(D_{i+1}^{24}(j+1))|_j$ and $Max(D_i^{out}(j+1))|_j = DB_i^{Max} + min(D_{i+1}^{24}(j+1))|_j$ is for the convenience of calculation only. Exact expressions will be discussed in a later section.

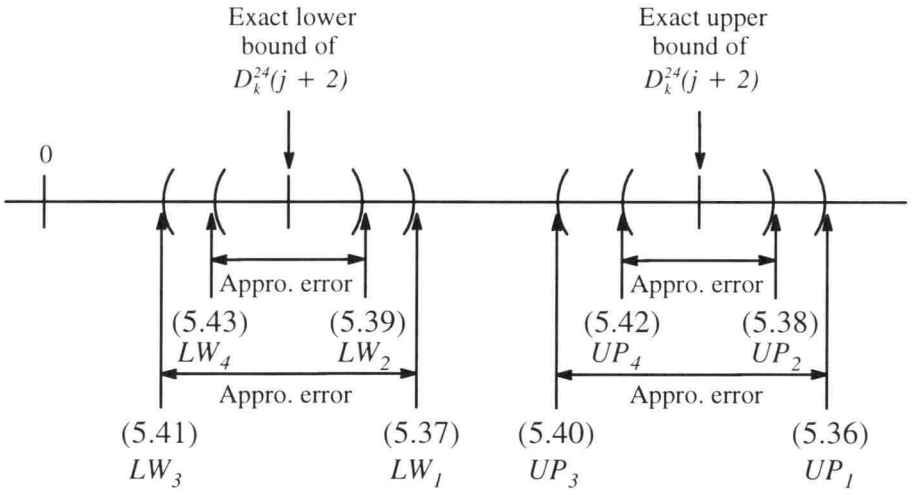
5.1.5 Several Approximations To $D_k^{in}(j+2)$

According to the definition of equivalent input delay $D_k^{in}(j+2) = DB_k(j+1) + D_k^{42}(j+2)$ and from the expression (5.3) of $D_k^{42}(j+2)$, we have

$D_k^{in}(j+2)$

$= Max(DB_k(j+1),$

$$DB_{k-1}(j+1) + \sum_{l=k}^{k}(F_l(j+2) - F_l(j+1)) - \sum_{l=k+1}^{k+1} D_l^{24}(j+1),$$



* If a micropipeline is initially reset, expressions (5.41) and (5.43) will coincide with exact lower bound.

Figure 5.4: Relative relationship among different approximations of $D_k^{24}(j + 2)$.

$$DB_{k-2}(j+1) + \sum_{l=k-1}^{k} (F_l(j+2) - F_l(j+1)) - \sum_{l=k}^{k+1} D_l^{24}(j+1),$$

$$\cdots\cdots$$

$$DB_1(j+1) + \sum_{l=2}^{k} (F_l(j+2) - F_l(j+1)) - \sum_{l=3}^{k+1} D_l^{24}(j+1),$$

$$DB_0(j+1) + \sum_{l=1}^{k} (F_l(j+2) - F_l(j+1)) - \sum_{l=2}^{k+1} D_l^{24}(j+1)) \tag{5.44}$$

where $0 \le k \le m$,

$$DB_0(j+1) = D_{in}(j+2).$$

If the stage–delay bounds are given and pipeline is initially reset, then

$$Max(D_k^{in}(j+2))|_j$$

$$= Max(DB_k^{Max},$$

$$DB_{k-1}^{Max} + \sum_{l=k}^{k} (F_l^{Max} - F_l^{min}),$$

$$DB_{k-2}^{Max} + \sum_{l=k-1}^{k} (F_l^{Max} - F_l^{min}),$$

$$\cdots\cdots$$

$$DB_1^{Max} + \sum_{l=2}^{k} (F_l^{Max} - F_l^{min}),$$

$$DB_0^{Max} + \sum_{l=1}^{k} (F_l^{Max} - F_l^{min}))$$

$$= UP_1(D_k^{in}(j+2))|_j \tag{5.45}$$

$$min(D_k^{in}(j+2))|_j$$

$$\le Max(DB_k^{min},$$

$$DB_{k-1}^{min} - \sum_{l=k}^{k} (F_l^{Max} - F_l^{min}),$$

$$DB_{k-2}^{min} - \sum_{l=k-1}^{k} (F_l^{Max} - F_l^{min}),$$

$$\cdots\cdots$$

$$DB_1^{min} - \sum_{l=2}^{k} (F_l^{Max} - F_l^{min}),$$

$$DB_0^{min} - \sum_{l=1}^{k} (F_l^{Max} - F_l^{min}))$$

$$= LW_1(D_k^{in}(j+2))|_j \tag{5.46}$$

Several conclusions can be reached from previous discussions.

$$1> \; Max(D_k^{42}(j+2))|_j = UP_1(D_k^{42}(j+2))|_j \tag{5.47}$$

$$min(D_k^{42}(j+2))|_j \leq LW_1(D_k^{42}(j+2))|_j \tag{5.48}$$

$$2> \; Max(D_k^{in}(j+2))|_j = UP_1(D_k^{in}(j+2))|_j \tag{5.49}$$

$$min(D_k^{in}(j+2))|_j \leq LW_1(D_k^{in}(j+2))|_j \tag{5.50}$$

$$3> \; Max(D_k^{in}(j+2))|_j \geq Max(D_k^{42}(j+2))|_j \tag{5.51}$$

$$min(D_k^{in}(j+2))|_j \geq min(D_k^{42}(j+2))|_j \tag{5.52}$$

4> Comparing expressions (5.21) and (5.45), we find

$$UP_1(D_k^{in}(j+2))|_j = \begin{cases} DB_k^{Max} & \text{if } UP_1(D_k^{42}(j+2))|_j = 0 \\ F_k^{Max} + B_k^{min} + UP_1(D_k^{42}(j+2))|_j & \text{otherwise} \end{cases} \tag{5.53}$$

and, comparing expressions (5.22) and (5.46), we have

$$LW_1(D_k^{in}(j+2))|_j = \begin{cases} DB_k^{min} & \text{if } LW_1(D_k^{42}(j+2))|_j = 0 \\ F_k^{min} + B_k^{Max} + LW_1(D_k^{42}(j+2))|_j & \text{otherwise} \end{cases} \tag{5.54}$$

Note that, the conditions *"0s"* in expressions (5.53) and (5.54) correspond to the first element instead of the calculation result (the other elements could be zeros after calculation) of expressions (5.21) and (5.22). In general,

$Max(D_k^{in}(j+2))|_j \neq F_k^{Max} + B_k^{Max} + Max(D_k^{42}(j+2))|_j$, and

$min(D_k^{in}(j+2))|_j \neq F_k^{min} + B_k^{min} + min(D_k^{42}(j+2))|_j$

If the representation of $D_k^{42}(j+2)$ in (5.19) is used to find $D_k^{in}(j+2)$, the representation of $D_k^{in}(j+2)$ becomes

$$
\begin{aligned}
&D_k^{in}(j+2) \\
&= Max(DB_k(j+1),\ min(\\
&\quad DB_{k-1}(j+1) + F_k(j+2) - F_k(j+1) + \sum_{l=k-1}^{k-1} D_l^{42}(j+1+k-l), \\
&\quad DB_{k-1}(j+1) + F_k(j+2) + B_k(j) - DF_{k+1}(j) + \sum_{l=k-1}^{k} D_l^{42}(j+1+k-l), \\
&\quad \cdots\cdots \\
&\quad DB_{k-1}(j+1) + F_k(j+2) + B_k(j) - \sum_{l=k+1}^{m-1} (B_l(j+1+k-l) - B_l(j+k-l)) \\
&\qquad\qquad - DF_m(j+1-m+k) + \sum_{l=k-1}^{m-1} D_l^{42}(j+1+k-l), \\
&\quad DB_{k-1}(j+1) + F_k(j+2) + B_k(j) - \sum_{l=k+1}^{m} (B_l(j+1+k-l) - B_l(j+k-l)) \\
&\qquad\qquad - DF_{m+1}(j-m+k) + \sum_{l=k-1}^{m} D_l^{42}(j+1+k-l))) \qquad (5.55)
\end{aligned}
$$

The conclusions are

$$1>\ Max(D_k^{42}(j+2))|_j \geq UP_3(D_k^{42}(j+2))|_j \qquad (5.56)$$

$$min(D_k^{42}(j+2))|_j \geq LW_3(D_k^{42}(j+2))|_j \qquad (5.57)$$

$$2>\ Max(D_k^{in}(j+2))|_j \geq UP_3(D_k^{in}(j+2))|_j \qquad (5.58)$$

$$min(D_k^{in}(j+2))|_j \geq LW_3(D_k^{in}(j+2))|_j \qquad (5.59)$$

$$3>\ Max(D_k^{in}(j+2))|_j \geq Max(D_k^{42}(j+2))|_j \qquad (5.60)$$

$$min(D_k^{in}(j+2))|_j \geq min(D_k^{42}(j+2))|_j \tag{5.61}$$

4>

$$UP_3(D_k^{in}(j+2))|_j = \begin{cases} DB_k^{Max} & if\ UP_3(D_k^{42}(j+2))|_j = 0 \\ F_k^{Max} + B_k^{min} + UP_3(D_k^{42}(j+2))|_j & otherwise \end{cases} \tag{5.62}$$

$$LW_3(D_k^{in}(j+2))|_j = \begin{cases} DB_k^{min} & if\ LW_3(D_k^{42}(j+2))|_j = 0 \\ F_k^{min} + B_k^{Max} + LW_3(D_k^{42}(j+2))|_j & otherwise \end{cases} \tag{5.63}$$

5.1.6 Several Approximations To $D_k^{out}(j+1)$

Similar to the previous derivation and according to the definition of equivalent output delay $D_k^{out}(j+1) = DF_k(j+1) + D_{k+1}^{24}(j+1)$, we have the following representation, if $D_{k+1}^{24}(j+2)$ is represented as the form of (5.2).

$$\begin{aligned} &D_k^{out}(j+2) \\ &= Max(DF_k(j+2), \\ &\quad DF_{k+1}(j+1) + B_k(j+2) - B_k(j+1) - \sum_{l=k}^{k} D_l^{42}(j+2+k-l), \\ &\quad DF_{k+2}(j) + \sum_{l=k+1}^{k+1} (B_l(j+2+k-l) - B_l(j+1+k-l)) + B_k(j+2) - B_k(j+1) \\ &\qquad - \sum_{l=k}^{k+1} D_l^{42}(j+2+k-l), \\ &\quad \cdots\cdots \\ &\quad DF_m(j+2-m+k) + \sum_{l=k+1}^{m-1} (B_l(j+2+k-l) - B_l(j+1+k-l)) + B_k(j+2) \\ &\qquad - B_k(j+1) - \sum_{l=k}^{m-1} D_l^{42}(j+2+k-l), \end{aligned}$$

$$DF_{m+1}(j+1-m+k)+\sum_{l=k+1}^{m}(B_l(j+2+k-l)-B_l(j+1+k-l))$$

$$+B_k(j+2)-B_k(j+1)-\sum_{l=k}^{m}D_l^{42}(j+2+k-l)) \quad (5.64)$$

The conclusions are

$$1>\ Max(D_{k+1}^{24}(j+2))|_j \le UP_1(D_{k+1}^{24}(j+2))|_j \quad (5.65)$$

$$min(D_{k+1}^{24}(j+2))|_j \le LW_1(D_{k+1}^{24}(j+2))|_j \quad (5.66)$$

$$2>\ Max(D_k^{out}(j+2))|_j \le UP_1(D_k^{out}(j+2))|_j \quad (5.67)$$

$$min(D_k^{out}(j+2))|_j \le LW_1(D_k^{out}(j+2))|_j \quad (5.68)$$

$$3>\ Max(D_k^{out}(j+2))|_j \ge Max(D_{k+1}^{24}(j+2))|_j \quad (5.69)$$

$$min(D_k^{out}(j+2))|_j \ge min(D_{k+1}^{24}(j+2))|_j \quad (5.70)$$

4>

$$UP_1(D_k^{out}(j+2))|_j=\begin{cases}DF_k^{Max} & \text{if } UP_1(D_{k+1}^{24}(j+2))|_j=0\\ B_k^{Max}+F_k^{min}+UP_1(D_{k+1}^{24}(j+2))|_j & \text{otherwise}\end{cases} \quad (5.71)$$

$$LW_1(D_k^{out}(j+2))|_j=\begin{cases}DF_k^{min} & \text{if } LW_1(D_{k+1}^{24}(j+2))|_j=0\\ F_k^{Max}+B_k^{min}+LW_1(D_{k+1}^{24}(j+2))|_j & \text{otherwise}\end{cases} \quad (5.72)$$

Note that, $UP_1(D_k^{out}(1)) = DF_k^{Max}$, and $LW_1(D_k^{out}(1)) = DF_k^{min}$.

If the representation of $D_{k+1}^{24}(j+2)$ in (5.18) is used to find $D_k^{out}(j+2)$, then $D_k^{out}(j+2)$ becomes

$D_k^{out}(j+2)$

$= Max(DF_k(j+2),\ min($

$$DF_{k+1}(j+1)+B_k(j+2)-B_k(j+1)+\sum_{l=k+2}^{k+2}D_l^{24}(j+1),$$

$$DF_{k+1}(j+1) + B_k(j+2) + F_k(j+1) - DB_{k-1}(j+1) + \sum_{l=k+1}^{k+2} D_l^{24}(j+1),$$

.....

$$DF_{k+1}(j+1) + B_k(j+2) + F_k(j+1) - \sum_{l=2}^{k-1}(F_l(j+2) - F_l(j+1)) - DB_1(j+1) + \sum_{l=3}^{k+2} D_l^{24}(j+1),$$

$$DF_{k+1}(j+1) + B_k(j+2) + F_k(j+1) - \sum_{l=1}^{k-1}(F_l(j+2) - F_l(j+1)) - DB_0(j+1) + \sum_{l=2}^{k+2} D_l^{24}(j+1))) \quad (5.73)$$

The conclusions are

$$1> \; Max(D_{k+1}^{24}(j+2))|_j \geq UP_3(D_{k+1}^{24}(j+2))|_j \quad (5.74)$$

$$min(D_{k+1}^{24}(j+2))|_j = LW_3(D_{k+1}^{24}(j+2))|_j \quad (5.75)$$

$$2> \; Max(D_k^{out}(j+2))|_j \geq UP_3(D_k^{out}(j+2))|_j \quad (5.76)$$

$$min(D_k^{out}(j+2))|_j = LW_3(D_k^{out}(j+2))|_j \quad (5.77)$$

$$3> \; Max(D_k^{out}(j+2))|_j \geq Max(D_{k+1}^{24}(j+2))|_j \quad (5.78)$$

$$min(D_k^{out}(j+2))|_j \geq min(D_{k+1}^{24}(j+2))|_j \quad (5.79)$$

4>

$$UP_3(D_k^{out}(j+2))|_j = \begin{cases} DF_k^{Max} & \text{if } UP_3(D_{k+1}^{24}(j+2))|_j = 0 \\ B_k^{Max} + F_k^{min} + UP_3(D_{k+1}^{24}(j+2))|_j & \text{otherwise} \end{cases} \quad (5.80)$$

$$LW_3(D_k^{out}(j+2))|_j = \begin{cases} DF_k^{min} & \text{if } LW_3(D_{k+1}^{24}(j+2))|_j = 0 \\ F_k^{Max} + B_k^{min} + LW_3(D_{k+1}^{24}(j+2))|_j & \text{otherwise} \end{cases} \quad (5.81)$$

Note that, $UP_3(D_k^{out}(1)) = DF_k^{Max}$, and $LW_3(D_k^{out}(1)) = DF_k^{min}$.

5.2 Design Procedure And Guidelines

In this section, how to design a linear micropipeline is shown based upon expressions (5.34) and (5.35), such that the resulting maximum and minimum output loop delays satisfy a specification. That is, given a delay bound, $Q^{min} \leq Q \leq Q^{Max}$, of a combinational logic circuit , how does one design a linear micropipeline such that its output loop delay $T^{l}_{m+1}(j+1)$ satisfies $T^{min} \leq T^{l}_{m+1}(j+1) \leq T^{Max}$? In the above statement, Q represents the original logic circuit delay before partition and Q^{min} and Q^{Max} symbolize the minimum and maximum signal delays, while T^{min} and T^{Max} designate the minimum and maximum output loop delay requirement after the original logic circuit has been partitioned and implemented using a linear micropipeline. For clarity, register propagation delays and C–element physical delays are ignored in this discussion without loss of generality. The effect of these two delays on performance of micropipelines will be discussed in Chapter 6. Backward delays are also ignored. Assuming that a m–stage micropipeline is considered: let $F_i^{Max} = a_i T^{Max}$ and $F_i^{min} = b_i T^{Max}$, $0 \leq i \leq m+1$, where $F_0^{Max} = D_{in}^{Max}$, $F_0^{min} = D_{in}^{min}$, $F_{m+1}^{Max} = D_{out}^{Max}$ and $F_{m+1}^{min} = D_{out}^{min}$. Since both register propagation delays and C–element physical delays are ignored, we temporarily assume $\sum_{l=1}^{m} F_l^{Max} = Q^{Max}$.

To satisfy the specification using fixed stage–delay micropipelines, we need

i> at least $\left\lceil \frac{Q^{Max}}{T^{Max}} \right\rceil$ pipe–stages if each stage–delay = T^{Max} except for a certain stage, if any (the situation when $\frac{Q^{Max}}{T^{Max}}$ is indivisible), which has the stage–delay = $Q^{Max} - \left\lfloor \frac{Q^{Max}}{T^{Max}} \right\rfloor * T^{Max}$; or

ii> at most $\left\lceil \frac{Q^{Max}}{T^{min}} \right\rceil$ pipe–stages if each stage–delay = T^{min} except for a certain stage, if any (the situation when $\frac{Q^{Max}}{T^{min}}$ is indivisible), which has the stage–delay = $Q^{Max} - \left\lfloor \frac{Q^{Max}}{T^{min}} \right\rfloor * T^{min}$.

The resulting output loop delay is $T^{l}_{m+1}(j+1) = T^{Max}$ for the former condition and is $T^{l}_{m+1}(j+1) = T^{min}$ for the latter condition.

If this circuit is implemented using a variable stage–delay micropipeline, we can rewrite expression (5.34) and (5.35) as

$$T^{l}_{m+1}(j+1) \geq b_{m+1}T^{Max} \tag{5.82}$$

$$\begin{aligned} T^{l}_{m+1}(j+1) \leq Max(&a_{m+1}T^{Max}, \\ &a_m T^{Max}, \\ &(\sum_{l=m-1}^{m} a_l - \sum_{l=m}^{m} b_l)T^{Max}, \\ &(\sum_{l=m-2}^{m} a_l - \sum_{l=m-1}^{m} b_l)T^{Max}, \\ &\cdots\cdots \\ &(\sum_{l=0}^{m} a_l - \sum_{l=1}^{m} b_l)T^{Max}) \end{aligned} \tag{5.83}$$

The goal is to find coefficients a_i and b_i, $0 \leq i \leq m+1$, such that expressions (5.82) and (5.83) satisfy requirement $T^{min} \leq T^{l}_{m+1}(j+1) \leq T^{Max}$. From $F_i^{Max} = a_i T^{Max}$ and $F_i^{min} = b_i T^{Max}$, we know

$$F_i^{Max} = a_i T^{Max} \geq F_i^{min} = b_i T^{Max} \Rightarrow a_i \geq b_i,\ 0 \leq i \leq m+1. \tag{5.84}$$

In general, there are two terms in each element (equation) of expression (5.35). The first and second elements have second term equal to 0. From expression (5.35) and the

upper bound requirement, we know the first term F_i^{Max}, $0 \le i \le m + 1$, in expression (5.35)

$$F_i^{Max} = a_i T^{Max} \le T^{Max} \Rightarrow a_i \le 1 \tag{5.85}$$

From conditions (5.84) and (5.85), we also know

$$a_i \ge b_i \text{ and } a_i \le 1 \Rightarrow b_i \le 1 \tag{5.86}$$

Also from our previous assumption

$$\sum_{l=1}^{m} F_l^{Max} = Q^{Max} \Rightarrow \sum_{l=1}^{m} a_l T^{Max} = Q^{Max} \tag{5.87}$$

From expression (5.82), to satisfy the lower bound requirement, we need

$$b_{m+1} T^{Max} \ge T^{min} \Rightarrow b_{m+1} \ge \frac{T^{min}}{T^{Max}} \tag{5.88}$$

From expression (5.83), to satisfy the upper bound requirement, coefficients of T^{Max} must be less than 1. That is,

$a_{m+1} \le 1$, and

$a_m \le 1$, and

$$\sum_{l=m-1}^{m} a_l - \sum_{l=m}^{m} b_l \le 1, \text{ and}$$

$$\sum_{l=m-2}^{m} a_l - \sum_{l=m-1}^{m} b_l \le 1, \text{ and}$$

. . . . . .

$$\sum_{l=0}^{m} a_l - \sum_{l=1}^{m} b_l \le 1. \tag{5.89}$$

There are many solutions that satisfy conditions (5.84) through (5.89). One of the possible solutions is obtained by replacing the inequality signs with equality signs in (5.88) and (5.89). With some simple algebraic manipulations, we get

$$b_{m+1} = \frac{T^{min}}{T^{Max}}, \text{ and}$$

$a_{m+1} = 1$, and

$a_m = 1$, and

$a_{i-1} = b_i = c_{i-1}, 1 \le i \le m$, which is substituted into equation (5.87), we get

$$(c_1 + c_2 + \cdots + c_{m-1} + 1)T^{Max} = Q^{Max} \tag{5.90}$$

From expression (5.84), we have

$$a_i \ge b_i \Rightarrow c_i \ge c_{i-1},\ 1 \le i \le m. \tag{5.91}$$

If Q^{Max} is a multiple of T^{Max} and if we let $c_1 = c_2 = \cdots = c_{m-1} = 1$, where $m = \frac{Q^{Max}}{T^{Max}}$, this is equivalent to the fixed stage–delay case with stage–delay = T^{Max} for each stage. The coefficients a_0 , b_0 and b_1 (note that a_0=b_1 in (5.90)) can be chosen arbitrarily as long as they satisfy conditions (5.84) through (5.86). If the number of pipe–stages m used to compose a pipeline is given or limited, the possible solutions will be reduced. Otherwise, the number of pipe–stage m can also be a controllable parameter when we design a micropipeline.

Once cs and m are determined, all lower bounds F_i^{min} and upper bounds F_i^{Max}, $0 \le i \le m + 1$, of stages become known, assuming that the bounds of (environmental) input and output delays are also controllable. It is possible that the resulting $\sum_{l=1}^{m} F_l^{min}$ is greater than, equal to or less than Q^{min} since no restriction like $\sum_{l=1}^{m} F_l^{min} = Q^{min}$ is imposed on this design procedure, as stated above. It is of no concern if $\sum_{l=1}^{m} F_l^{min} = Q^{min}$. Is there any problem if $\sum_{l=1}^{m} F_l^{min} \ne Q^{min}$? Before this question is answered, the effect of partitioning a combinational logic block is discussed first. Given a combinational logic circuit with delay bounds $Q^{min} \le Q \le Q^{Max}$, what is the relationship between $\sum_{l=1}^{m} Q_l^{min}$ and Q^{min}, and $\sum_{l=1}^{m} Q_l^{Max}$ and Q^{Max} if this combinational logic block is partitioned into m sub–blocks (stages), where Q_l^{min} is the delay lower bound and

Q_l^{Max} is the delay upper bound for stage l, $1 \le l \le m$? Without loss of generality, Figure 5.5(a) depicts a combinational logic block which is divided into three stages (m=3) as an example. The longest path within the original block is denoted as G which has delay equal to Q^{Max}, while the shortest path is designated as I which has delay equal to Q^{min}. Other paths within this block which have delays bounded by $[Q^{min},Q^{Max}]$ are generically represented as H. In this example, $\sum_{l=1}^{3} I_l = Q^{min}$ and $\sum_{l=1}^{3} G_l = Q^{Max}$. However, the individual stage–delay bounds are determined by choosing the maximum and minimum delays within each stage, i.e., $Q_l^{min} = minimum[G_l,H_l,I_l]$ and $Q_l^{Max} = Maximum[G_l,H_l,I_l]$, $1 \le l \le 3$. The condition $\sum_{l=1}^{3} Q_l^{min} = Q^{min}$ holds if and only if $minimum[G_l,H_l,I_l] = I_l$, $1 \le l \le 3$. Otherwise, for example, in Figure 5.5(a), if partition is made such that $minimum[G_1,H_1,I_1] = I_1$, $minimum[G_2,H_2,I_2] = H_2$ (implying $H_2 < I_2$), and $minimum[G_3,H_3,I_3] = I_3$, then $\sum_{l=1}^{3} Q_l^{min} = I_1 + H_2 + I_3 < I_1 + I_2 + I_3 = Q^{min}$. In summary, if a combinational logic block with upper and lower bounds is partitioned into m stages, the summation of the lower bounds of each stage will be less than or equal to the original undivided lower bound, i.e., $\sum_{l=1}^{m} Q_l^{min} \le Q^{min}$. A similar approach can be applied to obtain the upper bounds relationship. That is, the summation of the upper bounds of each stage will be greater than or equal to the original undivided upper bound, i.e., $\sum_{l=1}^{m} Q_l^{Max} \ge Q^{Max}$. A numerical example is demonstrated in Figure 5.5(b). The original combinational block is bounded by [*11,20*] but after partitioning the summation of lower bounds becomes *3+3+3=9* (*<11*) and the summation of upper bounds becomes *9+5+8=22* (*>20*).

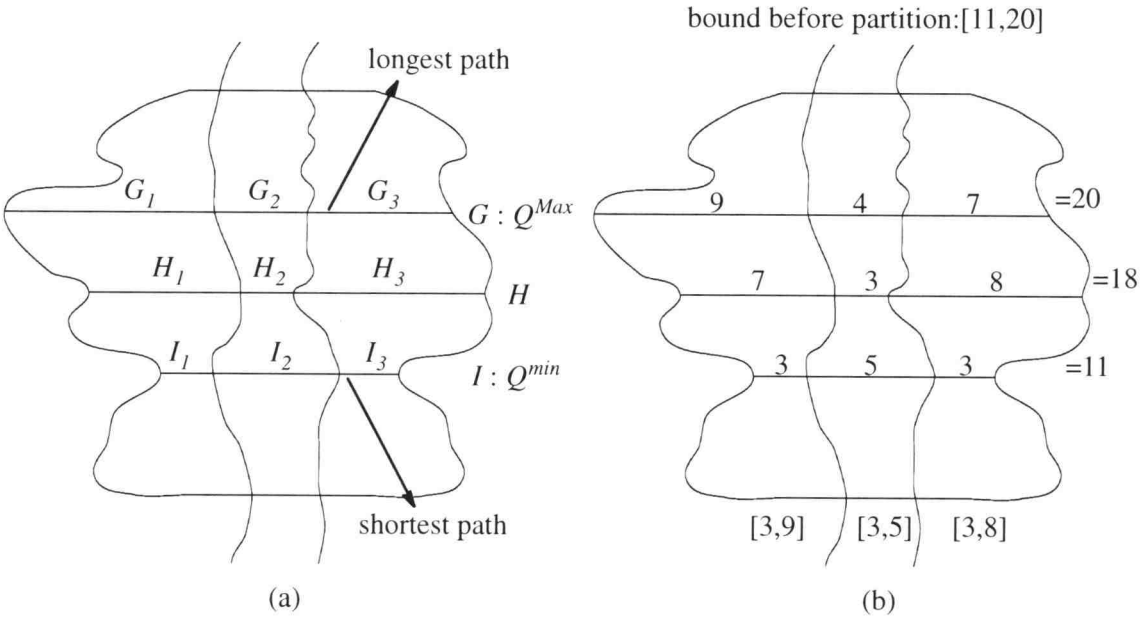


Figure 5.5: Partitioning a combinational logic block.
(a) A combinational logic block is arbitrarily partitioned into 3 stages.
(b) An example shows the delay bounds change before and after partition.

Continuing the discussion of our design solutions, will the condition $\sum_{l=1}^{m} F_l^{min} < Q^{min}$ cause a problem? From previous study, we know it is safe. How about if $\sum_{l=1}^{m} F_l^{min} > Q^{min}$? This indicates that the solution can not be implemented physically no matter how the original block is partitioned. There are three choices. First, we can find another set of coefficients such that $\sum_{l=1}^{m} F_l^{min} \leq Q^{min}$. The second choice is to leave the coefficients intact and really implement this circuit by artificially adding extra delays into each stage according to the coefficients we obtained. In fact, the condition $\sum_{l=1}^{m} F_l^{min} > Q^{min}$ relaxes the lower bounds at stages and provides additional design freedom. Therefore, the third choice is to redesign the stages with, for example, less hardware complexity by taking advantage of the relaxation in lower bounds. In the above design procedure, the assumption $\sum_{l=1}^{m} F_l^{Max} = Q^{Max}$ was made. In reality, this assumption is not necessary but reduces the possible solutions.

Now, if a linear micropipeline is given and does not satisfy the bounds requirement, how can we effectively and efficiently modify this micropipeline such that its output loop delay turns out to meet the requirement? Since a lower bound is easy to meet (by arbitrarily reducing the magnitude of D_{out}^{min}), we are more interested in how to reach upper bound requirement, i.e., how to reduce the maximum output loop delay. Four guidelines (or approaches) based upon expression (5.35) are addressed in this section. The first approach is to increase each stage's lower bound. This approach is obvious due to the "–" sign in front of the lower bound term F_l^{min} in expression (5.35). The second approach is to switch stages if allowed. The third approach is to shift some delays from one stage to the other. Finally, the fourth approach is to split a stage into

two or more stages. Usually, the above approaches can be combined to effectively and efficiently reduce the magnitude of expression (5.35). An example will be given to demonstrate these approaches in the following section. Among these approaches, the stage–splitting approach is most commonly used and is further discussed below.

Figure 5.6 illustrates the effect of stage splitting. Splitting a stage i in a micropipeline corresponds to splitting an equation (element) with its first term equal to DB_i^{Max} in expression (5.35). Two phenomena are warranted when a stage i is divided. First, the magnitude of the original equation is larger than the magnitude of each of the split equations. That is, in Figure 5.6, the magnitude of $DB_i^{Max} + \sum_{l=i+1}^{m} (F_l^{Max} - F_l^{min})$ is larger than the magnitude of any equations within the shaded area. Second, the magnitudes of the other equations (corresponding to unsplit stages) in expression (5.35) remain unchanged. The above two phenomena are based upon the assumption of $DB_i^{Max} = \sum_{l=1}^{n} DB_{i,l}^{Max}$ and $DB_i^{min} = \sum_{l=1}^{n} DB_{i,l}^{min}$, where n is the number of stages into which stage i is split and $DB_{i,l}^{Max}$ and $DB_{i,l}^{min}$ represent maximum and minimum total stage–delays for new split stages. Given these phenomena, which stages should be divided remains to be answered. Our experience in synchronous designs suggests that the stage with the largest maximum total stage–delay DB_i^{Max} is a good candidate. Surprisingly, based upon our approach, this is not the case in asynchronous pipelines. The stages with corresponding magnitudes in expression (5.35) exceeding the required upper bound are all candidates for further division. Since a larger total stage–delay does not necessarily imply a larger corresponding magnitude in expression (5.35), splitting a stage with the largest maximum total stage–delay may not reduce the value of the maximum upper bound. Generally speaking, splitting inappropriate stages won't change the resulting value of expression (5.35). For example, in Figure 5.6, if $DB_k^{Max} > DB_i^{Max}$ but

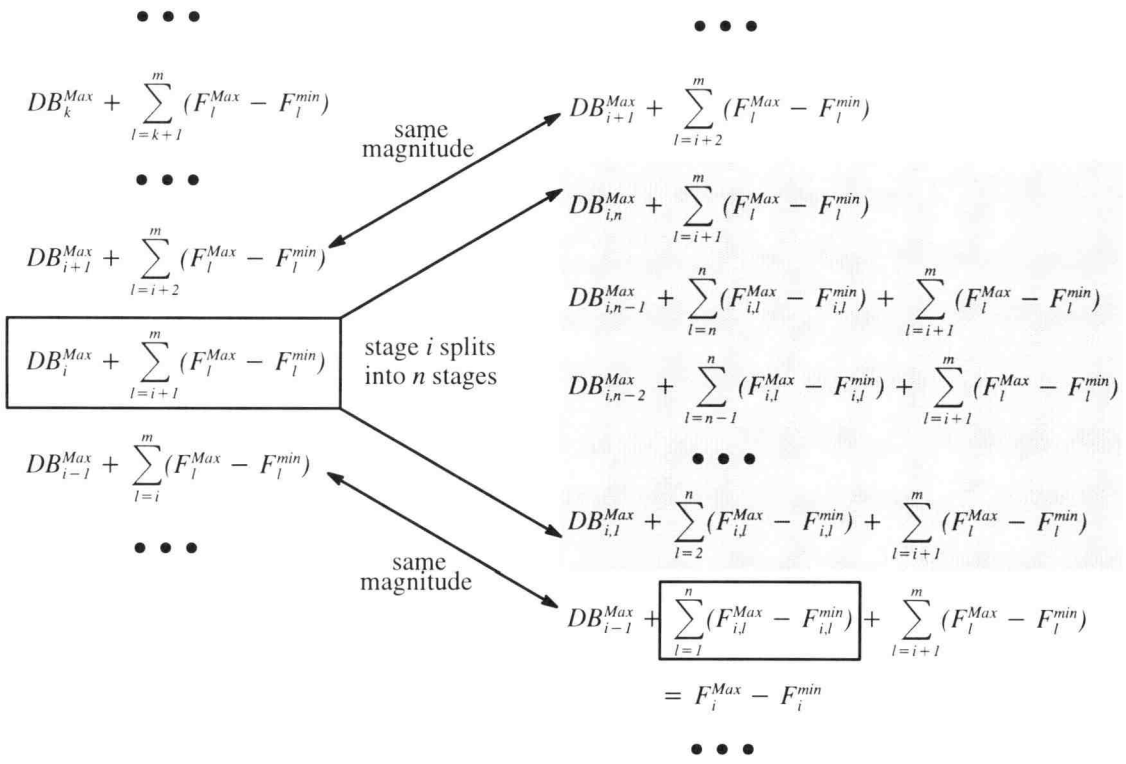


Figure 5.6: The equations corresponding to stage i being split into n stages.

$DB_i^{Max} + \sum_{l=i+1}^{m} (F_l^{Max} - F_l^{min}) > T^{Max} > DB_k^{Max} + \sum_{l=k+1}^{m} (F_l^{Max} - F_l^{min})$, splitting stage k won't change the final value of expression (5.35). However, by splitting stage i, a design that meets the requirement may be obtained.

How do we know if a stage–splitting approach will help to satisfy the upper bound? By setting the first term of elements whose magnitudes exceed the upper bound in expression (5.35) to zero, if the magnitudes of resulting equations still cannot meet the requirement, stage–splitting does not help in reducing the upper bound at all. That is, if $\sum_{l=1}^{k}(F_l^{Max} - F_l^{min}) > T^{Max}$, $1 \le k \le m$, for any k, the expression (5.35) will never be satisfied by simply using a stage–splitting approach. In other words, if the environmental input and output delays can not be split, then stage–splitting helps only if $D_{out}^{Max} \le T^{Max}$ and $D_{in}^{Max} + \sum_{l=1}^{m}(F_l^{Max} - F_l^{min}) \le T^{Max}$. All of these criteria are based on the truth that the first term in (5.35) can be reduced by stage–splitting but the second term cannot.

If the goal is to obtain a linear micropipeline whose output loop delay satisfies the bound $T^{min} \le T_{m+1}^{1}(j+1) \le T^{Max}$, then it seems that the implementation of fixed stage–delay micropipelines can meet this goal very well and efficiently, compared with the implementation of a variable stage–delay micropipeline. This is because, not only are the number of pipe–stages ($\left\lceil \frac{Q^{Max}}{T^{Max}} \right\rceil$) needed to implement a circuit less, but also the hardware costs less. Most variable stage–delay micropipelines use a dual–rail encoding method for which hardware costs double those of fixed stage–delay micropipelines. Moreover, fixed stage–delay micropipelines are even easier to implement. Then, why do we need to even consider implementing variable stage–delay implementation? Because the average throughput, the other performance design metric, is different. Fixed stage–delay micropipelines have fixed throughput rates, but variable stage–delay micro-

pipelines have variable throughput rates. As will be seen in Chapter 6, variable stage–delay micropipelines may have better average throughput than fixed stage–delay micropipelines. In real circuits design, both throughput bounds and average throughput should be satisfied.

5.3 Numerical Example

In this section, a numerical design example is given to demonstrate the design procedure stated in the previous section. Several approaches to helping the existing design meet the specification are also included in this section.

5.3.1 Original Design

Consider a combinational logic circuit having delay bounds expressed as $10 \le Q \le 46$. Design a three–stage micropipeline such that its output loop delay have bounds $1 \le T_4^l(j+1) \le 31$, assuming backward delay $B_i(j+1) = 0, 1 \le i \le 3$, $j \ge 0$.

From the design specification, we have $Q^{min} = 10$, $Q^{Max} = 46$, $T^{min} = 1$, $T^{Max} = 31$ and $m = 3$. The design goal is to find coefficients a_i and b_i, $0 \le i \le 4$. From (5.87), we have

$$\sum_{l=1}^{3} a_l * 31 = 46 \Rightarrow a_1 + a_2 + a_3 = \frac{46}{31}$$

From (5.88), we have

$$b_4 \ge \frac{1}{31}$$

From (5.89), we have

$$a_4 \le 1$$

$$a_3 \le 1$$

$$\sum_{l=2}^{3} a_l - \sum_{l=3}^{3} b_l \leq 1$$

$$\sum_{l=1}^{3} a_l - \sum_{l=2}^{3} b_l \leq 1$$

$$\sum_{l=0}^{3} a_l - \sum_{l=1}^{3} b_l \leq 1$$

One set of coefficients that satisfies the above constraints is

$$a_4 = \frac{2}{31},\ a_3 = \frac{15}{31},\ a_2 = \frac{18}{31},\ a_1 = \frac{13}{31},\ \text{and}\ a_0 = \frac{3}{31}$$

$$b_4 = \frac{1}{31},\ b_3 = \frac{6}{31},\ b_2 = \frac{9}{31},\ b_1 = \frac{7}{31},\ \text{and}\ b_0 = \frac{2}{31}$$

Note that the value of b_0 is arbitrarily assigned as long as it is less than or equal to a_0.

Thus, the bounds of each stage in the resulting micropipeline are

$$2 \leq D_{in}(j+2) \leq 3,$$

$$7 \leq F_1(j+1) \leq 13,$$

$$9 \leq F_2(j+1) \leq 18,$$

$$6 \leq F_3(j+1) \leq 15, \text{ and}$$

$$1 \leq D_{out}(j+1) \leq 2.$$

Substituting these bounds into expression (5.34) and (5.35), We have

$$T_4^l(j+1) \geq 1, \text{ and}$$

$$\begin{aligned} T_4^l(j+1) = Max(&2, \\ &15, \\ &18 + (15 - 6), \\ &13 + (15 - 6) + (18 - 9), \\ &3 + (15 - 6) + (18 - 9) + (13 - 7)) \\ = Max(&2, 15, 27, 31, 27) = 31. \end{aligned}$$

Clearly, this design satisfies the specification $1 \le T_4^l(j+1) \le 31$ (equivalently, $\frac{1}{31} \le P(j+1) \le 1$).

5.3.2 Design Modification

Now consider a change in the design requirement. The upper bound of the new specification is changed from *31* to *29*, that is, $\frac{1}{29} \le P(j+1) \le 1$. We can either go through the whole design procedure and find a new solution (coefficient set) that satisfies the new requirement, or we can use the following approaches based upon the old solution to reduce the maximum output loop delay.

Approach 1>: stage splitting

Although F_2^{Max} *(18)* is the largest upper bound of stage–delays in the micropipeline, splitting the second stage won't change the maximum output loop delay *(31)*. This is because the magnitude of elements corresponding to the second stage is *27,* which is less than *31*. Instead, the stage corresponding to having a magnitude of elements equal to *31* should be split. In our example, the first stage should be split. If we divide it into two stages such that

$$7 \le F_1(j+1) \le 13 \Rightarrow 2 \le F_{1,2}(j+1) \le 7, \text{ and}$$

$$5 \le F_{1,1}(j+1) \le 6$$

The new maximum output loop delay becomes

$$T_4^l(j+1) \le Max(2, 15, 27, 25, 29, 27) = 29.$$

Approach 2>: delay shifting

If we shift two units of the first stage's lower bounds to the second stage's lower bounds, we then have

$$5 \le F_1(j+1) \le 13 \text{ and } 11 \le F_2(j+1) \le 18$$

The resulting maximum output loop delay becomes

$$T_4^l(j + 1) \leq Max(2, 15, 27, 29, 27) = 29.$$

Approach 3>: lower bound increasing

By increasing the second stage's lower bound to 11, that is,

$$11 \leq F_2(j + 1) \leq 18$$

the resulting maximum output loop delay changes to

$$T_4^l(j + 1) \leq Max(2, 15, 27, 29, 25) = 29.$$

Approach 4>: stage switching

If we switch the first and second stages, such that

$$9 \leq F_1(j + 1) \leq 18 \text{ and } 7 \leq F_2(j + 1) \leq 13$$

the resulting maximum output loop delay becomes

$$T_4^l(j + 1) \leq Max(2, 15, 27, 28, 27) = 28.$$

All of the above approaches lead to smaller maximum output loop delays. These approaches can be combined to reduce the maximum output loop delay even more efficiently and effectively. During the process of modifying a design, there are many choices in each of the approaches. Some choices result in a smaller maximum output loop delay. Others cause it to grow larger. Each of the above approaches may have a possible side–effect, which is making the average throughput smaller (or making the average output loop delay larger). The best approaches or choices to adopt are those that will make both average and maximum output loop delays meet the specifications.

5.4 Summary

In this chapter, performance–related issues of a linear micropipeline with variable stage–delays are discussed. Given the lower and upper bounds of each stage's forward and backward delays, the micropipeline's lower and upper throughput bounds can be easily obtained. The lower output loop delay bound is a sufficient condition for a micro-

pipeline to meet its lower specification; that is, the "=" sign might not occur under certain conditions of combinational forward and backward delays for each stage. The upper output loop delay bound, however, is a sufficient and necessary condition. Therefore, we are able to find a micropipeline's worst performance (exact value). Based upon representations of throughput bounds, a design procedure is introduced and several design approaches are suggested to help achieve a new bound for an existing design (apparently, there are ample solutions). Each design approach may have possible side–effects, making the average throughput smaller or the average output loop delay larger. The best approaches or choices to adopt are those that will make both average and maximum output loop delays meet the specifications.

6. PERFORMANCE ISSUES FOR SYNCHRONOUS VS. ASYNCHRONOUS PIPELINES

Given a pipeline circuit, two types of performance parameter, average throughput and throughput bounds, are usually concerned. In synchronous pipelines, average throughput is equal to the throughput bounds (upper bound equal to lower bound) which are limited to the worst stage–delay. As to the asynchronous pipelines (micropipelines), however, their behavior is quite different. Since throughput bounds (related to the overall output loop–delay bounds) of asynchronous pipelines have already been discussed, in this chapter average throughput will be focused and will be compared to the performance of synchronous pipelines. Instead of adopting the approach from the probability point of view [31,32], we are interested in using numerical forms to explain some interesting phenomena based upon the model and expressions derived from previous chapters. Note that the average throughput (or average output loop delay) discussed in Sections 6.3 and 6.4 is the "average" performance over the *limited* number of input data set, therefore, the conclusions drawn in these sections can not be directly applied to the case of *infinite* number of input data set.

The main expressions used for performance comparison in this chapter are (5.1) and (5.2), which are restated in the following.

$$
\begin{aligned}
T^{l}_{m+1}(j+1) = Max(&DF_{m+1}(j+1),\\
&DB_{m}(j+1),\\
&DB_{m-1}(j+1) + \sum_{l=m}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=m+1}^{m+1} D_l^{24}(j+1),\\
&DB_{m-2}(j+1) + \sum_{l=m-1}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=m}^{m+1} D_l^{24}(j+1),\\
&\ldots\ldots
\end{aligned}
$$

$$DB_1(j+1) + \sum_{l=2}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=3}^{m+1} D_l^{24}(j+1),$$

$$DB_0(j+1) + \sum_{l=1}^{m}(F_l(j+2) - F_l(j+1)) - \sum_{l=2}^{m+1} D_l^{24}(j+1)) \quad (5.1)$$

$D_k^{24}(j+2)$

$= Max(0,$

$$DF_k(j+1) - DB_{k-1}(j+1) - \sum_{l=k-1}^{k-1} D_l^{42}(j+1+k-l),$$

$$DF_{k+1}(j) + \sum_{l=k}^{k}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{k} D_l^{42}(j+1+k-l),$$

. . . . . .

$$DF_m(j+1-m+k) + \sum_{l=k}^{m-1}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m-1} D_l^{42}(j+1+k-l),$$

$$DF_{m+1}(j-m+k) + \sum_{l=k}^{m}(B_l(j+1+k-l) - B_l(j+k-l)) - DB_{k-1}(j+1)$$

$$- \sum_{l=k-1}^{m} D_l^{42}(j+1+k-l)) \quad (5.2)$$

Generally speaking, each element (except for the first and the second) within expression (5.1) consists of three terms. The first term $DB_i(j+1)$ (including environmental input and output delay), representing the sum of forward and backward delays, is called *delay–sum* (or *backward total stage–delay* compared to *forward total stage–delay* $DF_i(j+1)$). The second term $F_l(j+2) - F_l(j+1)$ expressing the difference of forward delay is called *forward delay slope.* As defined before, the third term $D_l^{24}(j+1)$

is logic delay. The last two terms of (5.1) only occur in asynchronous circuits and are the results of handshaking. Hence, they are called *asynchronous (*or *handshaking)* parameters.

6.1 Performance Constraints

We will first compare the performance of a synchronous pipeline with that of a fixed stage–delay asynchronous pipeline. This comparison is based upon the usage of edge–triggered registers — rising (or falling) edge–triggered registers for synchronous circuits and double–edge–triggered registers [29] for asynchronous circuits. The reason why double–edge–triggered registers are used in asynchronous circuits is because a micropipeline uses a two–phase handshaking scheme. Details on how clock skew affects clock period in synchronous pipelines can be found elsewhere [33]. Some restrictions are briefly summarized below.

$$2T_{skew,max} \leq T_{reg} + T_{change} - T_{hold} \tag{6.1}$$

$$T_{period} \geq T_{reg} + T_{settle} + T_{setup} + 2T_{skew,max} \tag{6.2}$$

where T_{period} : clock period

$T_{skew,max}$: maximum clock skew

T_{reg} : register propagation delay

T_{hold} : register hold time

T_{setup} : register setup time

T_{change} : the time for the first output bit of the combinational logic to change

T_{settle} : combinational logic settle time

Expression (6.1) is required to prevent same input signal from being latched by two consecutive registers at the same clock edge (with phase shifted due to clock skew). The clock period should satisfy condition (6.2) in order to latch valid (stable) data.

On the right–hand side graph of Figure 3.4(a), physical delay d is lumped with delays in both data–token and space–token paths. To clearly demonstrate performance constraint, it is better to separate d from delays, as shown in Figure 6.1, i.e. $F_i = F_i^{'} + d$ and $B_i = B_i^{'} + d$. From Figure 6.1, there are two constraints for micropipelines to work properly, as shown below.

$$B_i^{'} + d + T_{reg} + T_{change} \geq T_{hold} \tag{6.3}$$

$$F_i^{'} + d \geq T_{reg} + T_{settle} + T_{setup} \tag{6.4}$$

To prevent new data from overriding old data which have not been latched by the next register, a relationship like (6.3) should be enforced. Expression (6.4) must be satisfied to avoid the violation of the bundled data convention for micropipelines. Expression (6.3) and (6.4) are based upon the assumption that both edge–triggered and double edge–triggered registers have the same T_{reg}, T_{hold} and T_{setup}.

Usually $B_i^{'}$ is zero for simple micropipelines, as we will assume here. In asynchronous pipelines, expression (6.3) can easily be satisfied in practice. However, it is difficult to satisfy expression (6.1) in synchronous pipelines. This is because, as die size grows and transistor feature size continues to shrink, wiring delays play a more and more important role compared with gate delays. That is, clock skew (phase delay due to long wiring) may be longer than T_{reg} and/or T_{change} and may cause expression (6.1) to fail.

From expression (6.2), the throughput of a synchronous pipeline (best case) is $1/T_{period} = 1/(T_{reg} + T_{settle} + T_{setup} + 2T_{skew,max})$ compared to $1/(F_i + B_i) = 1/(T_{reg} + T_{settle} + T_{setup} + d)$ for an asynchronous pipeline. The relative value of $2T_{skew,max}$ and d determines if asynchronous pipelines are better than, equal to or worse than their synchronous counterparts. If $2T_{skew,max} \approx d$ is assumed, the circuits implemented using micropipelines [9] have comparable performance to the same circuits implemented using synchronous pipelines.

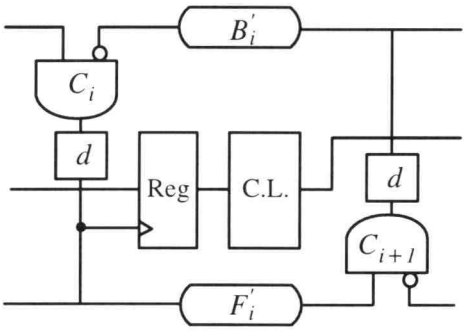


Figure 6.1: One stage of a micropipeline.

6.2 Output Loop Delay < Worst Stage–Delay

It is commonly believed that asynchronous pipelines can achieve better performance (average) than synchronous pipelines. The main difference between them is that in synchronous circuits next data cannot be processed until clock has arrived; whereas, in asynchronous circuits next data can be *potentially* executed right after the completion of current data processing. Therefore, the performance of synchronous circuits is bounded by worst–case delay of a combinational logic block; whereas, the performance of asynchronous circuits is bounded by "average" delay. Usually, the "average" sense only comes from the worst–case and best–case physical delays. For example, for a non–pipelined n–bit ripple–carry adder, the time needed to process some data that need no carry ripple (best case) is much less than the time required to process certain data that require n carry propagation (worst case). Therefore, the "average" propagation time over a large set of data is in between. In this section, it is pointed out that the "average" sense as stated above is not thorough, especially when a pipeline is considered. More factors affecting "average" performance will be described in this section.

Consider, for example, a simple pipeline where backward delay, $B_i(j+1)$, is ignored. If the input and output environmental delays can be made arbitrarily small compared to $F_i(j+1)$, $1 \le i \le m$, the first and the last elements in expression (5.1) won't possibly be the maximum value. Moreover, if the rest of the elements in expression (5.1) have the same probability (this assumption is, in general, not true) of being selected as the representation of output loop delay, the expectation of output loop delay (considering delay–sum only) for a m–stage pipeline is the sum of the product of each stage–delay's mean and probability. That is,

$$E[T^1_{m+1,delay-sum}] = \sum_{i=1}^{m} \frac{1}{m} E[F_i] = \frac{1}{m} \sum_{i=1}^{m} E[F_i] \tag{6.5}$$

where $E[T^1_{m+1,delay-sum}]$: expectation (mean) of delay–sum term of output loop

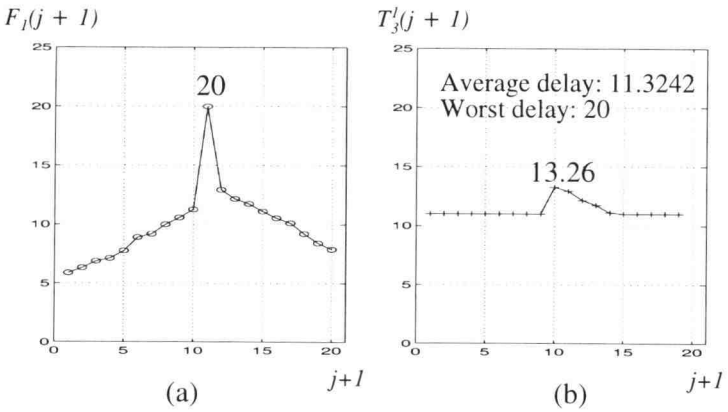
delay

$E[F_i]$: expectation (mean) of stage–delay for stage i

$\frac{1}{m}$: probability of each element selected in expression (5.1)

It appears that equation (6.5) matches the "average" sense for asynchronous pipelines. However, equation (6.5) is only one of the many factors that make asynchronous pipelines faster. From expression (5.1), we also notice that forward delay slope and logic delay may also affect the performance of asynchronous pipelines. Apparently, forward delay slope can be positive, zero or negative. If it is negative, output loop delay will be reduced. This implies that not only the delay at instant *j*, but also the delay difference (or delay pattern) between adjacent *j*s, affects performance. Since logic delay is never less than zero, this term always contributes positively toward obtaining better performance, based on expression (5.1). However, if expression (5.17) is considered, logic delay makes the performance of asynchronous pipelines worse. Both statements are correct. It is the approach (left–hand side or right–hand side) that makes these two statements look opposite.

Figure 6.2 shows the results of a two–stage pipeline simulation for a case in which average output loop delay (11.3242) is less than the maximum stage–delay (20). In order to simplify our observation and to clarify the simulation result, only the delay of the first stage in this micropipeline is set to be variable, as shown in Figure 6.2(a) (assuming $B_1(j+1) = 0$ for all *j*s), while other delays are set to be constant, i.e., D_{in}=9, D_2=4 and D_{out}=11. Figure 6.2(c) lists the specific situations when *j*=9 and 10, in which the output loop delay is $T_3^1(j+1) = DB_1(j+1) - D_3^{24}(j+1)$ according to expression (5.1) for m=2. Interestingly, comparing Figure 6.2(a) and Figure 6.2(b), it is noticed that the maximum stage–delay (20) won't be reflected to the output loop delay with the same magnitude. That is, even if a micropipeline stage has a larger delay at a given instant j=k, the output loop delay could be less than this delay at the corresponding *j*. For



j	9	10
$B_1(j+1)+F_1(j+2)$	20	12.92
$F_1(j+2)-F_1(j+1)$	8.74	–7.08
$D_2^{24}(j+1)$	0	0
$D_3^{24}(j+1)$	6.74	0
Output Loop Delay	13.26	12.92

(c)

Figure 6.2: Simulation results with D_{in}=9,D_2=4 and D_{out}=11.
(a)Delay pattern of $F_1(j+1)$(assuming $B_1(j+1)$=0).
(b)Output loop delay $T_3^1(j+1)$.
(c)Several delay values when j=9 and 10.

example, $T_3^1(10) = DB_1(10) - D_3^{24}(10) = 20 - 6.74 = 13.26$ when j=9. We speculate that only part of the maximum stage–delay is passed (13.26) to the output (or environment) and the rest of it (20–13.26=6.74) is "absorbed" by its handshaking C–element. The amount of delay (6.74) absorbed by C–element is not detectable by the environment, hence, leading to a better performance. This kind of performance behavior does not occur in synchronous pipelines.

From the above discussion, we demonstrated that both the delay–sum term and the asynchronous parameters in expression (5.1) influence the performance of asynchronous pipelines. Considering the effect of the asynchronous parameters gives an added incentive to employ the asynchronous design approach over the synchronous approach when negative values of the asynchronous parameters are guaranteed. This is because the output loop delay becomes the delay–sum term minus the magnitudes of the asynchronous parameter, making "average" sense of delay even smaller. A different approach to describing the same result presented in this section can be found in [34].

6.3 Average Output Loop Delay > Worst Stage–Delay

Several reasons that give asynchronous pipelines better performance than synchronous pipelines have been explored and brought up. Since in practical applications the number of input data is finite, we will address the next question based on this assumption: do asynchronous pipelines always perform better than their synchronous counterparts? Again a two–stage linear micropipeline is investigated for clarity. The output loop delay representation is simplified by assuming $D_{in}(j + 2)$, $DB_2(j + 1)$ and $D_{out}(j + 1)$ to be constants and backward delay $B_1(j + 1) = 0$ for all js. Figure 6.3 shows the simulation result with D_{in}=30.8, D_2=4 and D_{out}=15. From Figure 6.3(a) and Figure 6.3(c), it is found that the average output loop delay (31.4300) happens to be larger than the maximum stage–delay (31). Apparently, reviewing expression (5.1), we

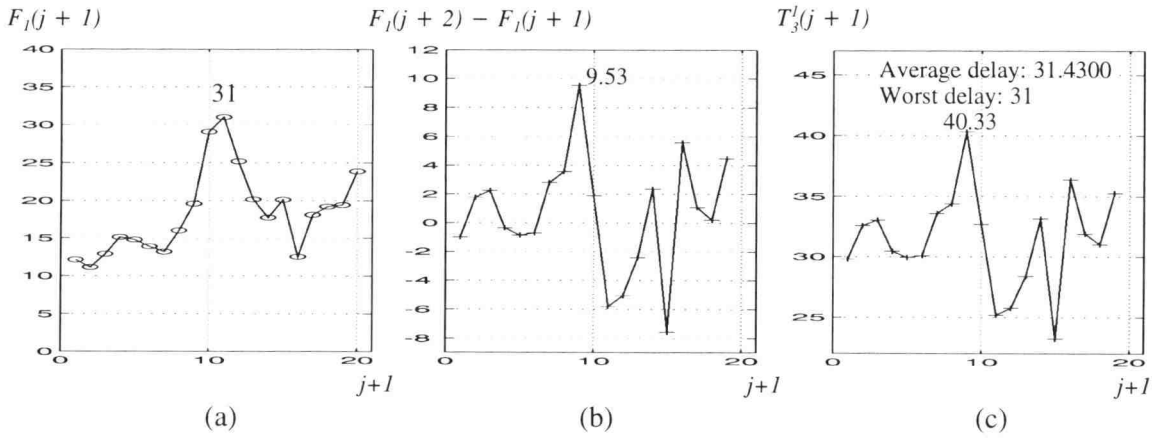


Figure 6.3: Simulation result with D_{in} =30.8, D_2=4 and D_{out}=15.
(a) Delay pattern of $F_1(j+1)$.
(b) Forward delay slope $F_1(j+2) - F_1(j+1)$.
(c) Output loop delay $T_3^1(j+1)$.

notice that this is caused by the overall effect of positive forward delay slope $F_1(j + 2) - F_1(j + 1)$, as shown in Figure 6.3(b). For example, when j=8, from the simulation result we found that $D_2^{24}(9)=D_3^{24}(9)=0$ (not shown in Figure 6.3) and $T_3^1(9) = D_{in} + (F_1(10) - F_1(9)) - (D_2^{24}(9) + D_3^{24}(9)) = 30.8 + 9.53 - 0 = 40.33$ (by trial and error, we found this equation in expression (5.1) satisfies our simulation result). The positive forward delay slope 9.53 when j=8 makes the output loop delay corresponding to the same j larger than the maximum stage–delay by the amount of 9.33 (= 40.33 – 31). From this simulation, we observed that the average output–loop delay for asynchronous pipelines may be larger than maximum stage–delay for a limited number of input data set. That is, asynchronous pipelines may have worse average performance than synchronous pipelines, given a set of input data.

6.4 Effect Of Delay Sequence (Pattern)

The discussions in sections 6.2 and 6.3 verify the statement that both the delay–sum and the asynchronous parameters affect the performance of asynchronous pipelines. Moreover, it has been shown that average output loop delay could be less than, greater than or equal to (obviously, this is possible although not mentioned in previous sections) the maximum stage–delay, given a set of input data. In other words, when the number of input data is finite, asynchronous pipelines could have better, worse or equal performance than synchronous pipelines, depending on the magnitude of each stage–delay and its pattern in time. The simulation result in Figure 6.4 shows how the delay pattern of forward delay $F_1(j + 1)$ and backward delay $B_1(j + 1)$ affects the output loop delay. This simulation is done for a two–stage micropipeline with D_{in}=12.8, D_2=4 and D_{out}=11. In the left most graph of Figure 6.4, six different cases of output loop delays are plotted against time. The right–hand side graphs of Figure 6.4 show the corresponding different cases of delay pattern (first stage). The delay elements (corresponding to

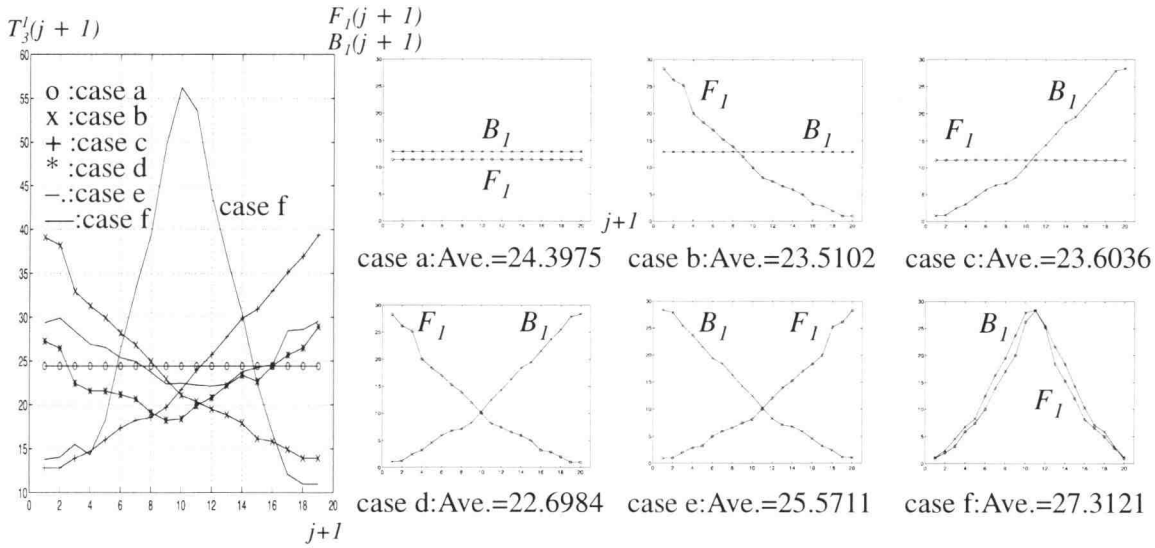


Figure 6.4: Simulation result with D_{in}=12.8, D_2=4 and D_{out}=11 for different cases of forward and backward delays sequence (pattern).

different j) of $F_l(j + 1)$ and $B_l(j + 1)$ for all these cases (except for cases *a, b* and *c,* where either $F_l(j + 1)$ or $B_l(j + 1)$ or both have constant values equal to the average of these delay elements) are the same except that their orderings are different.

Comparing average output loop delay for case *a* and case *b,* we observed that decreasing $F_l(j + 1)$ monotonously leads to better overall performance. On the contrary, by comparing case *a* and case *c,* we found that increasing $B_l(j + 1)$ monotonously makes circuits faster. Case *d* shows that the average output loop delay is made even smaller by decreasing $F_l(j + 1)$ and increasing $B_l(j + 1)$ monotonously. The opposite slopes of $F_l(j + 1)$ and $B_l(j + 1)$ applied to this pipeline, as shown in case *e*, slow down circuit operation. Among these simulations, case *f* gives the worst performance.

In a certain sense, we can explain, from expression (5.1) and (5.2), why case *d* gives the best performance among these cases. As discussed in section 6.2, negative forward delay slope and positive logic delay will potentially make output loop delay smaller. Decreasing $F_l(j + 1)$ monotonously means a negative forward delay slope which implies that a better performance might be achieved. If the term $B_l(j + 1 + k - l) - B_l(j + k - l)$ in expression (5.2), called *backward delay slope,* is positive, a more positive logic delay is achieved. Increasing $B_l(j + 1)$ monotonously guarantee it. One needs to note that decreasing and increasing monotonously the forward and backward delays are not practical in designing real circuits. The main purpose of this section is to show that the delay pattern does affect the average output loop delay of asynchronous pipelines. In comparison, there is no such effect in synchronous pipelines. One also should note that the three terms in expression (5.1) are not independent from each other. Hence, expression (5.1) should not be used as a guideline to design a high–performance (average throughput) asynchronous circuit. Establishing an algorithm to arrive at an optimized design (best performance) for an asynchronous pipeline is not trivial. This is because the delay pattern in each stage is non–deterministic. The study of asynchro-

nous pipelines from the probability point of views might be a good start in formalizing design guidelines to obtain the optimized average throughput [31,32].

6.5 Effect Of Number Of Stages

The effect of delay–sum, forward delay slope and logic delay on the performance of asynchronous pipelines has been presented in previous sections. However, from expression (5.1), there remain one more parameter which may affect performance. This is m, the number of stages for a pipeline. An interesting example in demonstrating how the number of pipe–stages may affect performance is given here. As we all know that the number of identical stages cascaded to form a synchronous pipeline does not change the overall pipeline's performance (ignoring clock skew). Is the same statement applicable to asynchronous pipelines? Figure 6.5(b) shows the output loop delays of a two–stage and a three–stage asynchronous pipelines with each stage's forward and backward delays for both pipelines shown in Figure 6.5(a). Apparently, from our simulation result with $D_{in} = 12.3$ and $D_{out} = 11.5$, these two pipelines have different output loop delays. This is because the m determines the number of summation terms having effect on performance in expression (5.1). For example, if the last element in expression (5.1) is chosen, then

for two–stage pipeline:

$$T_3^1(j+1) = D_{in} + \sum_{l=1}^{2}(F_l(j+2) - F_l(j+1)) - \sum_{l=2}^{3} D_l^{24}(j+1) \text{ , and}$$

for three–stage pipeline:

$$T_4^1(j+1) = D_{in} + \sum_{l=1}^{3}(F_l(j+2) - F_l(j+1)) - \sum_{l=2}^{4} D_l^{24}(j+1)$$

$T_4^1(j+1)$ contains two more terms, $F_3(j+2) - F_3(j+1)$ and $D_4^{24}(j+1)$, than $T_3^1(j+1)$, leading to a different output loop delay. Moreover, the values of $D_2^{24}(j+1)$

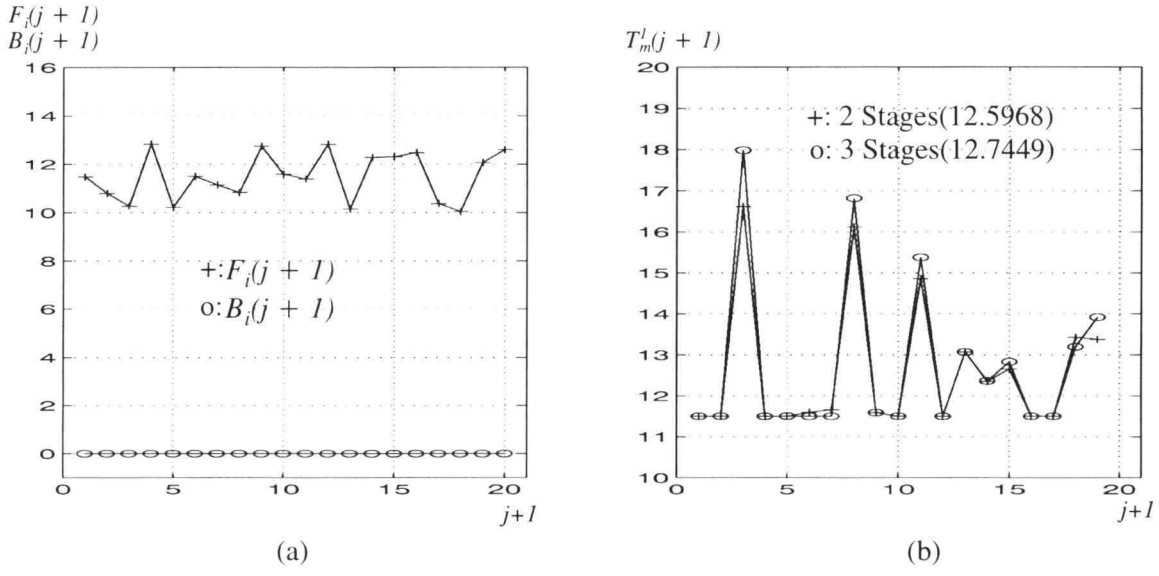


Figure 6.5: Simulation result with D_{in} =12.3 and D_{out}=11.5.
(a)Delay pattern of $F_i(j+1)$ and $B_i(j+1)$ used for all stages.
(b)Output loop delays for both 2–stage and 3–stage micropipelines.

and $D_3^{24}(j+1)$ in $T_3^1(j+1)$ are different from those in $T_4^1(j+1)$. Note that the results shown in Figure 6.5 does not include the effects of C–element physical delay and register propagation delay.

6.6 Summary

The main purpose for this chapter is to demonstrate the performance difference between synchronous and asynchronous pipelines. The conclusion drawn in this chapter for a fixed stage–delay micropipeline is that it has a constant performance. This performance is equal to the maximum stage–delay within the whole micropipeline. Moreover, its performance is equivalent to that of a synchronous pipeline as long as $2T_{skew,max} = d$, where d is the physical delay of a C–element. As for the performance of variable stage–delay micropipelines, we showed that not only stage–delay (delay–sum) but also stage–delay pattern (asynchronous parameters — forward delay slope and logic delay) affect performance. This leads to the conclusion that asynchronous pipelines may have better, worse or equivalent performance (average) than synchronous pipelines, given a finite number of input data set. The results in this chapter also show that the number of identical stages cascaded will affect the resulting performance.

7. PERFORMANCE EVALUATION OF TWO–DIMENSIONAL ASYNCHRONOUS PIPELINES

Linear micropipelines are also called one–dimensional micropipelines. The performance of this kind of pipeline for both fixed stage–delay and variable stage–delay cases has been studied in previous chapters. If the pipeline is initially reset, the upper bound of output loop delay calculated using our approach is tight (exact) for both cases. In this chapter the performance of two–dimensional micropipelines, based upon the linear pipeline model, is discussed. The two–dimensional pipelines presented in this chapter are constructed from all of the modules depicted in Chapter 3 Figure 3.5.

Since micropipelines feature modularity and composibility in circuit construction, we would like to adopt this feature as much as possible in evaluating their performance. This feature will reduce the complexity in calculating performance when considering a larger system. To facilitate the calculation while maintaining the accuracy of approximation, the Equal loop–delay theorem is used to derive performance estimations for each basic module. The result is a set of difference equations. Then, the given stage–delay bounds are applied to this difference equations to obtain the approximate upper and lower bounds. When evaluating the performance of a system constructed from these basic modules, the results (in terms of stage–delay bounds) obtained for each basic module can be applied directly to the system, instead of constructing huge expressions (difference equations) describing the system in terms of j.

It seems that the concept of equivalent input and output delays is not so important when linear pipelines are discussed. One of the reasons is because the output loop delay can be obtained directly using Eq.(5.1) without utilizing the concept of equivalent input and output delays. Of course, the same result can be reached if approached using equivalent input and output delays. As will be seen in the next chapter, the concept of equivalent input/output delay becomes more important when a two–dimensional pipeline sys-

tem is treated. As a result, the expressions for equivalent input and output delays for each module will be presented.

7.1 Performance Of Asynchronous Pipelines With Fork

A Fork pipeline, formed by Event AND module, is shown in Figure 7.1(a). The equivalent input and output delays ($D^{in}_{i-1}(j+2)$, $D^{out}_{q}(j+1)$ and $D^{out}_{p}(j+1)$) in this Fork figure are used to represent environmental input and output delays. The performance of Fork pipelines is developed using equivalent delay technique, as shown in Figure 7.1(b). The result can be applied to Fork pipelines with any stages. Note that to save space, we are only interested in upper bound calculations.

The equivalent input delay $D^{in}_{i,q}(j+2)$ can be represented as follows.

$$D^{in}_{i,q}(j+2) = D^{qp}_{a}(j+1) + B_i(j+1) + D^{42}_{i}(j+2) + F_i(j+2) \tag{7.1}$$

It should be noted that $D^{qp}_{a}(j+1)$ and $D^{42}_{i}(j+2)$ are not independent. Instead, their relationship is governed by an equation obtained by applying Equal loop–delay theorem to C_i. That is,

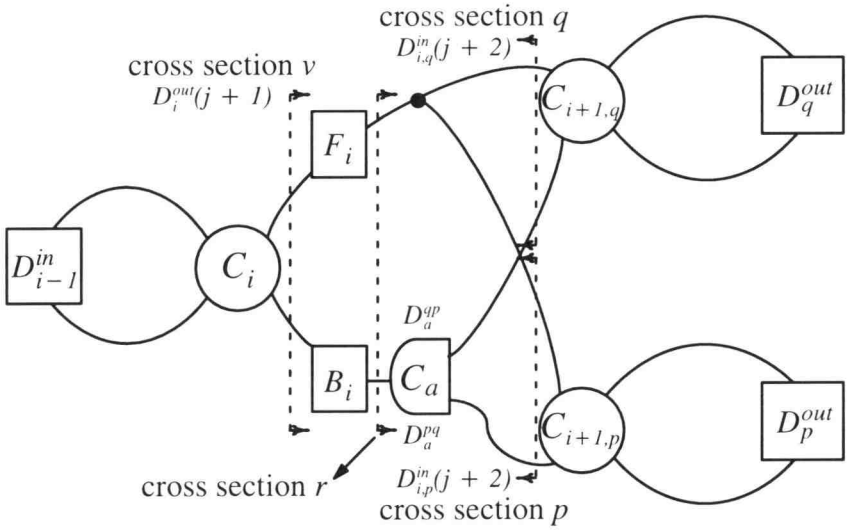
$$\begin{aligned} & D^{in}_{i-1}(j+2) + D^{24}_{i}(j+2) \\ &= F_i(j+1) + D^{24}_{q}(j+1) + D^{qp}_{a}(j+1) + B_i(j+1) + D^{42}_{i}(j+2) \qquad (7.2) \\ &\Rightarrow D^{42}_{i}(j+2) \\ &= Max(0,\ D^{in}_{i-1}(j+2) - F_i(j+1) - D^{24}_{q}(j+1) - D^{qp}_{a}(j+1) - B_i(j+1)) \end{aligned}$$

Therefore,

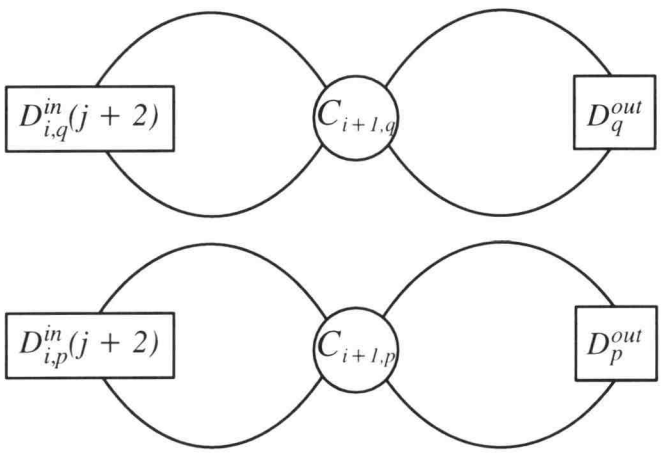
$$\begin{aligned} & D^{qp}_{a}(j+1) + D^{42}_{i}(j+2) \\ &= Max(D^{qp}_{a}(j+1),\ D^{in}_{i-1}(j+2) - F_i(j+1) - D^{24}_{q}(j+1) - B_i(j+1)) \qquad (7.3) \end{aligned}$$

Note that $D^{24}_{q}(j+1)$ is a shorthand of $D^{24}_{i+1,q}(j+1)$.

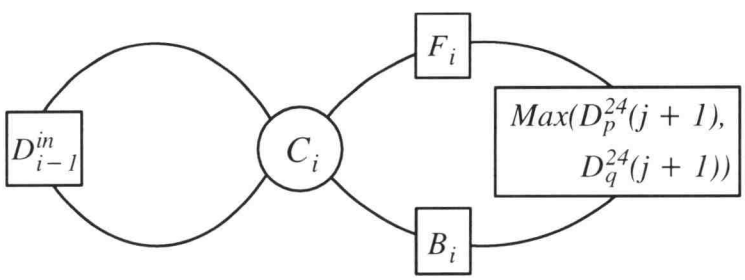
By applying Equal loop–delay theorem to Event AND C_a in Figure 7.1(a), we have



(a)



(b)



(c)

Figure 7.1: A two–dimensional micropipeline — Fork.
(a) General representation of pipelines with Fork.
(b) Equivalent circuit when looking into cross sections q and p.
(c) Equivalent circuit when looking into cross section r.

$$B_i(j+1) + D_i^{42}(j+2) + F_i(j+2) + D_q^{24}(j+2) + D_a^{qp}(j+2)$$

$$= B_i(j+1) + D_i^{42}(j+2) + F_i(j+2) + D_p^{24}(j+2) + D_a^{pq}(j+2)$$

$$D_a^{qp}(j+2) + D_q^{24}(j+2) = D_a^{pq}(j+2) + D_p^{24}(j+2)$$

$$\Rightarrow D_a^{qp}(j+2) = Max(0, D_p^{24}(j+2) - D_q^{24}(j+2)) \leq Max(0, D_P^{24}(j+2)) \quad (7.4)$$

$$D_a^{pq}(j+2) = Max(0, D_q^{24}(j+2) - D_p^{24}(j+2)) \leq Max(0, D_q^{24}(j+2)) \quad (7.5)$$

Note that $D_a^{qp}(1) = D_a^{pq}(1) = 0$ if the pipeline is initially reset. Also note that all delays in Figure 7.1 are variable.

From expression (7.4),

$D_a^{qp}(j+2) \leq Max(0, D_P^{24}(j+2))$ and

$$D_p^{24}(j+2) \leq Max(0,$$

$$D_p^{out}(j+1) - DB_i(j+1) - D_a^{pq}(j+1) - logic\ delay)$$

$$\leq Max(0,$$

$$D_p^{out}(j+1) - DB_i(j+1) - D_a^{pq}(j+1))$$

$$\leq Max(0,$$

$$D_p^{out}(j+1) - DB_i(j+1)) \quad (7.6)$$

$$\Rightarrow D_a^{qp}(j+2) \leq Max(0,$$

$$D_p^{out}(j+1) - DB_i(j+1)) \quad (7.7)$$

$$\Rightarrow Max(D_a^{qp}(j+2))|_j \leq Max(0,$$

$$D_p^{out,Max} - DB_i^{min}) \quad (7.8)$$

Considering (7.1), (7.3) and the initial condition where $D_q^{24}(1) = D_a^{qp}(1) = 0$, we have

$$D_{i,q}^{in}(2) = Max(B_i(1) + F_i(2),$$

$$D_{i-1}^{in}(2) - F_i(1) + F_i(2))$$

$$\Rightarrow Max(D_{i,q}^{in}(2))|_{j=0} = Max(F_i^{Max} + B_i^{Max},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \quad (7.9)$$

From (7.1), (7.3) and (7.7), we have

$$D_{i,q}^{in}(j+3) = D_{a}^{qp}(j+2) + B_i(j+2) + D_i^{42}(j+3) + F_i(j+3)$$

$$\leq Max(F_i(j+3) + B_i(j+2),$$

$$D_p^{out}(j+1) - F_i(j+2) + F_i(j+3) - B_i(j+1) + B_i(j+2),$$

$$D_{i-1}^{in}(j+3) + F_i(j+3) - F_i(j+2))$$

$$\Rightarrow Max(D_{i,q}^{in}(j+3))|_j \leq Max(F_i^{Max} + B_i^{Max},$$

$$D_p^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \tag{7.10}$$

Combining expressions (7.9) and (7.10), we conclude

$$Max(D_{i,q}^{in}(j+2))|_j \leq Max(F_i^{Max} + B_i^{Max},$$

$$D_p^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min})$$

$$= UP_1(D_{i,q}^{in}(j+2))|_j \tag{7.11}$$

And, the output loop delay will be

$$Max(T_{i+1,q}^{l}(j+1))|_j \leq Max(D_q^{out,Max},\ UP_1(D_{i,q}^{in}(j+2))|_j) \tag{7.12}$$

Similarly, we have

$$Max(D_{i,p}^{in}(j+2))|_j \leq Max(F_i^{Max} + B_i^{Max},$$

$$D_q^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min})$$

$$= UP_1(D_{i,p}^{in}(j+2))|_j \tag{7.13}$$

$$Max(T_{i+1,p}^{l}(j+1))|_j \leq Max(D_p^{out,Max},\ UP_1(D_{i,p}^{in}(j+2))|_j) \tag{7.14}$$

It is useful to obtain the equivalent output delay by looking into cross section *v*, as indicated in Figure 7.1(a), when the system becomes more complicated, as will be

seen in the next chapter. The equivalent output delay of $D_i^{out}(j+1)$ can be represented as follows.

$$D_i^{out}(j+1) = F_i(j+1) + D_q^{24}(j+1) + D_a^{qp}(j+1) + B_i(j+1), \text{ or} \tag{7.15}$$

$$D_i^{out}(j+1) = F_i(j+1) + D_p^{24}(j+1) + D_a^{pq}(j+1) + B_i(j+1) \tag{7.16}$$

$$\Rightarrow D_i^{out}(1) = F_i(1) + B_i(1)$$

$$\Rightarrow Max(D_i^{out}(1))|_{j=0} = F_i^{Max} + B_i^{Max} \tag{7.17}$$

From (7.15), we have

$$D_i^{out}(j+2) = F_i(j+2) + D_q^{24}(j+2) + D_a^{qp}(j+2) + B_i(j+2) \tag{7.18}$$

Since $D_a^{qp}(j+2) = Max(0, D_p^{24}(j+2) - D_q^{24}(j+2))$ from (7.4), we have

$$D_q^{24}(j+2) + D_a^{qp}(j+2) = Max(D_q^{24}(j+2),\ D_p^{24}(j+2)) \tag{7.19}$$

The above expression implies that only maximum logic delay ($D_q^{24}(j+2)$ or $D_p^{24}(j+2)$) can be detected when looking into cross section *r*. That is, logic delay $D_a^{qp}(j+2)$ or $D_a^{pq}(j+2)$ can not be detected. This property makes a system easier to analyze and the approximation result is more accurate. If the initial condition is also taken into account, the equivalent circuit can be viewed as shown in Figure 7.1(c). Due to the symmetry of Fork circuits and from (7.6), (7.18) and (7.19), we have

$$\begin{aligned} D_i^{out}(j+2) \le Max(&F_i(j+2) + B_i(j+2),\\ &D_p^{out}(j+1) - B_i(j+1) + B_i(j+2),\\ &D_q^{out}(j+1) - B_i(j+1) + B_i(j+2)) \end{aligned} \tag{7.20}$$

$$\begin{aligned} \Rightarrow Max(D_i^{out}(j+2))|_j \le Max(&F_i^{Max} + B_i^{Max},\\ &D_p^{out,Max} + B_i^{Max} - B_i^{min},\\ &D_q^{out,Max} + B_i^{Max} - B_i^{min}) \end{aligned} \tag{7.21}$$

Combining (7.17) and (7.21), we conclude

$$Max(D_i^{out}(j+1))|_j \le Max(F_i^{Max} + B_i^{Max},$$

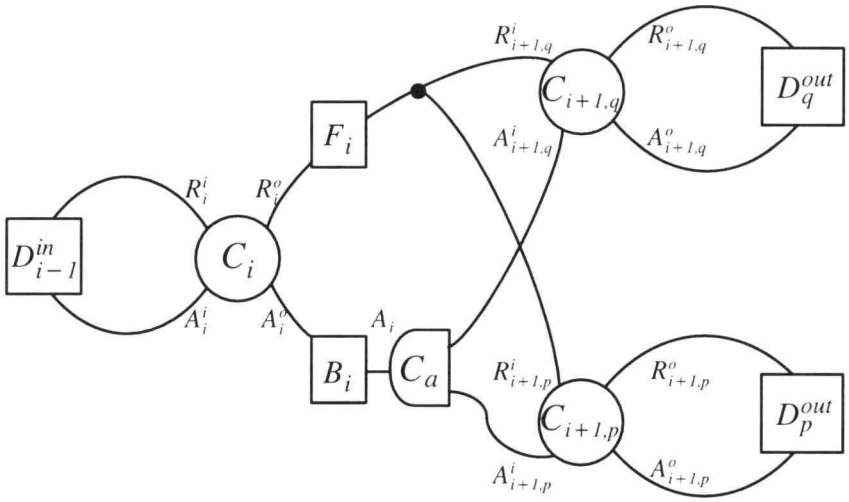
$$D_p^{out,Max} + B_i^{Max} - B_i^{min},$$

$$D_q^{out,Max} + B_i^{Max} - B_i^{min}) \tag{7.22}$$

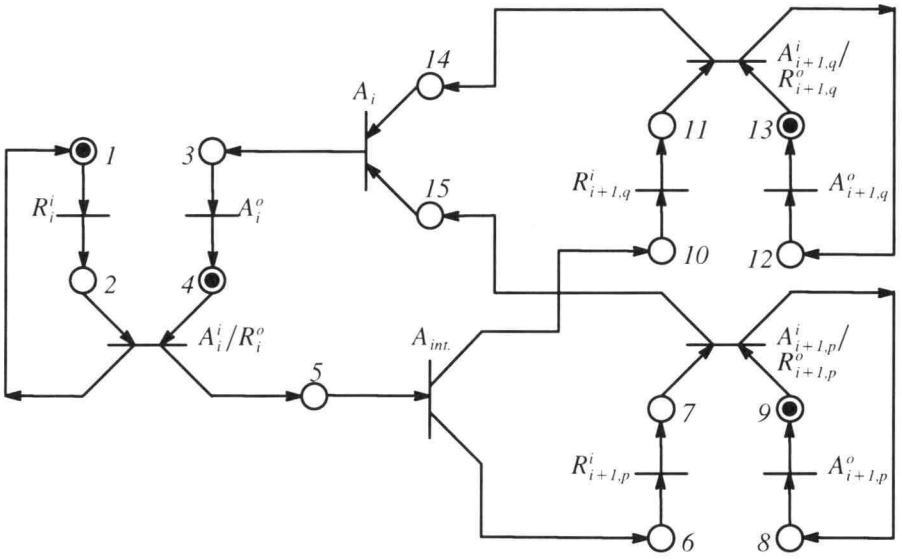
As was done for linear micropipelines, we would like to compare the output loop delays ($UP_{1,q}$ and $UP_{1,p}$) based on our approach with the exact upper bounds (*Max,q* and *Max,p*) obtained using CTSE. Figure 7.2(a) shows the Fork circuit and Figure 7.2(b) depicts its corresponding Petri net model. One dummy transition ($A_{int.}$) and two dummy places (*6* and *10*) are artificially added to the Petri net to model the fork behavior (one input token generates two output tokens). To correctly model performance, the delays of these two dummy places are set to zero. All of the C–element physical delays are assumed to be zero. Since only the upper bound is concerned in most of the applications, the simulation results in Table 7.1. show upper bound comparisons only. As observed from this table, our approaches have greater or equal values than the exact upper bound, which is as expected. Figure 7.2(c) shows this comparison based upon Table 7.1.

Table 7.1: Comparison of our approximations with the exact bounds using a general Fork pipeline as an example.

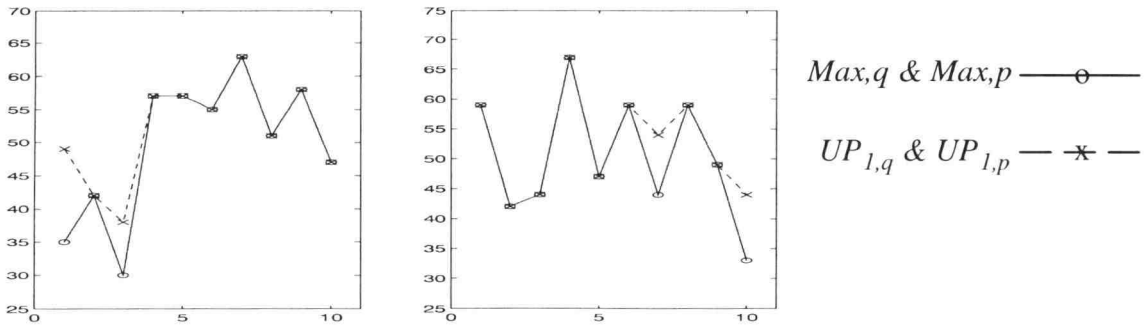
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_{i-1}^{in}	[2 15]	[26 40]	[2 15]	[26 30]	[26 30]	[6 10]	[2 15]	[6 30]	[16 35]	[2 15]
F_i	[1 10]	[20 22]	[1 10]	[20 22]	[20 22]	[10 11]	[1 13]	[10 31]	[20 31]	[1 10]
B_i	[3 18]	[4 15]	[3 8]	[4 15]	[4 15]	[4 7]	[3 10]	[14 17]	[14 17]	[3 8]
D_q^{out}	[29 35]	[12 14]	[25 30]	[20 54]	[20 34]	[10 55]	[30 35]	[10 35]	[10 35]	[25 30]
D_p^{out}	[21 25]	[12 14]	[21 24]	[22 44]	[22 44]	[22 44]	[39 44]	[22 24]	[22 44]	[31 33]
Max,q	35	42	30	57	57	55	63	51	58	47
$UP_{1,q}$	49	42	38	57	57	55	63	51	58	47
Max,p	59	42	44	67	47	59	44	59	49	33
$UP_{1,p}$	59	42	44	67	47	59	54	59	49	44



(a)



(b)



Max,q and $UP_{1,q}$ *Max,p* and $UP_{1,p}$

(c)

Figure 7.2: A Fork pipeline and its Petri net model.
(a) General representation of a Fork pipeline.
(b) Petri net model.
(c) Result comparison.

7.2 Performance of Asynchronous Pipelines with Join

A Join pipeline, merging two pipelines into one by using Event AND module, is shown in Figure 7.3(a). The equivalent input and output delays in this Join figure are used to represent the other possible circuits. We will develop the performance of Join pipelines using the equivalent delay technique. The result can be applied to Join pipelines with any stages.

The equivalent input delay $D_i^{in}(j+2)$ can be represented as follows.

$$D_i^{in}(j+2) = B_i(j+1) + D_{i,q}^{42}(j+2) + D_a^{qp}(j+2) + F_i(j+2), \text{ or} \tag{7.23}$$

$$D_i^{in}(j+2) = B_i(j+1) + D_{i,p}^{42}(j+2) + D_a^{pq}(j+2) + F_i(j+2) \tag{7.24}$$

By applying the Equal loop–delay theorem to Event AND C_a in Figure 7.3(a), we have

$$F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) + D_{i,q}^{42}(j+2) + D_a^{qp}(j+2)$$

$$= F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) + D_{i,p}^{42}(j+2) + D_a^{pq}(j+2)$$

$$\Rightarrow D_a^{qp}(j+2) + D_{i,q}^{42}(j+2) = D_a^{pq}(j+2) + D_{i,p}^{42}(j+2)$$

$$\Rightarrow D_a^{qp}(j+2) = Max(0, D_{i,p}^{42}(j+2) - D_{i,q}^{42}(j+2)) \le Max(0, D_{i,P}^{42}(j+2)) \tag{7.25}$$

$$D_a^{pq}(j+2) = Max(0, D_{i,q}^{42}(j+2) - D_{i,p}^{42}(j+2)) \le Max(0, D_{i,q}^{42}(j+2)) \tag{7.26}$$

Note that both $D_a^{qp}(1)$ and $D_a^{qp}(1)$ are not equal to zero in general and they will affect exact throughput bounds. Their values can be obtained by finding the time difference for the first token (one at each pipe branch) to travel from environment to the input ends of C–element C_a. Also note that all of the delays in Figure 7.3 are variable delays.

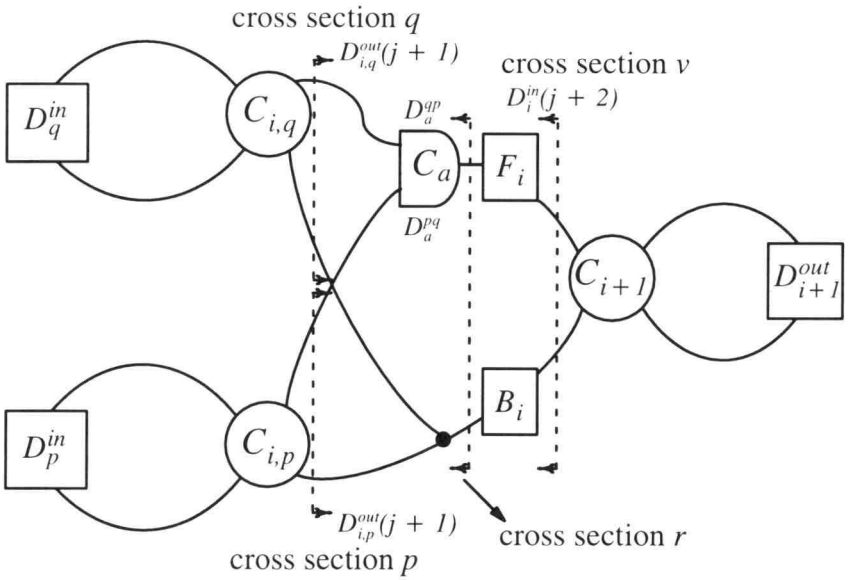
From (7.25) and (7.26), we have

$$D_a^{qp}(j+2) = Max(0, D_{i,p}^{42}(j+2) - D_{i,q}^{42}(j+2))$$

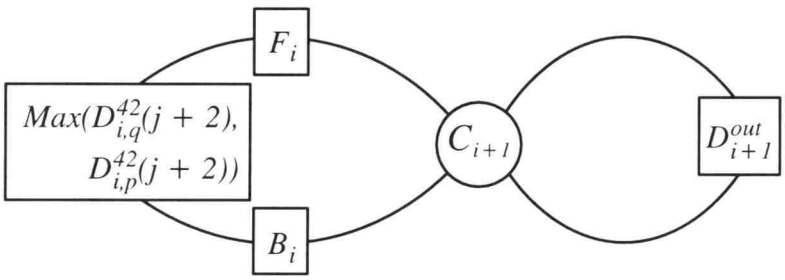
$$\Rightarrow D_a^{qp}(j+2) + D_{i,q}^{42}(j+2) = Max(D_{i,q}^{42}(j+2), D_{i,p}^{42}(j+2)) \tag{7.27}$$

$$D_a^{pq}(j+2) = Max(0, D_{i,q}^{42}(j+2) - D_{i,p}^{42}(j+2))$$

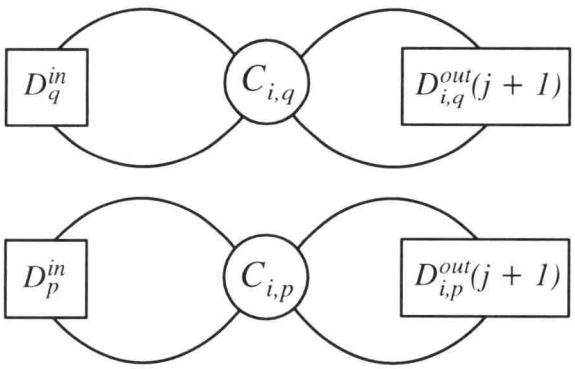
$$\Rightarrow D_a^{pq}(j+2) + D_{i,p}^{42}(j+2) = Max(D_{i,p}^{42}(j+2), D_{i,q}^{42}(j+2)) \tag{7.28}$$



(a)



(b)



(c)

Figure 7.3: A two–dimensional micropipeline — Join.
(a) General representation of pipelines with Join.
(b) Equivalent circuit when looking into cross section r.
(c) Equivalent circuit when looking into cross sections q and p.

Therefore, if we look into cross section r in Figure 7.3(a), the equivalent delay that will be seen is $Max(D^{42}_{i,q}(j+2), D^{42}_{i,p}(j+2))$, as shown in Figure 7.3(b). Based on expressions (7.27) and (7.28), the equivalent input delay $D^{in}_i(j+2)$ represented by expressions (7.23) and (7.24) can be re–written as

$$D^{in}_i(j+2) = B_i(j+1) + Max(D^{42}_{i,q}(j+2), D^{42}_{i,p}(j+2)) + F_i(j+2) \tag{7.29}$$

If we are able to find the delay bounds for either (7.27) or (7.28), the delay bounds for $D^{in}_i(j+2)$ can be obtained. By applying the Equal loop–delay theorem to C–element $C_{i,p}$, we have

$$\begin{aligned} D^{42}_{i,p}(j+2) \leq\ & Max(0, \\ & \quad D^{in}_p(j+2) - DF_i(j+1) - D^{pq}_a(j+1) - logic\ delay) \\ \leq\ & Max(0, \\ & \quad D^{in}_p(j+2) - DF_i(j+1) - D^{pq}_a(j+1)) \\ \leq\ & Max(0, \\ & \quad D^{in}_p(j+2) - DF_i(j+1)) \end{aligned} \tag{7.30}$$

Similarly, we have

$$\begin{aligned} D^{42}_{i,q}(j+2) \leq\ & Max(0, \\ & \quad D^{in}_q(j+2) - DF_i(j+1) - D^{qp}_a(j+1)) \\ \leq\ & Max(0, \\ & \quad D^{in}_q(j+2) - DF_i(j+1)) \end{aligned} \tag{7.31}$$

Combining expressions (7.29), (7.30) and (7.31), we conclude

$$\begin{aligned} D^{in}_i(j+2) \leq\ & Max(F_i(j+2) + B_i(j+1), \\ & \quad D^{in}_q(j+2) + F_i(j+2) - F_i(j+1), \\ & \quad D^{in}_p(j+2) + F_i(j+2) - F_i(j+1)) \\ \Rightarrow Max(D^{in}_i(j+2))|_j \leq\ & Max(DB^{Max}_i, \end{aligned}$$

$$D_q^{in,Max} + F_i^{Max} - F_i^{min},$$

$$D_p^{in,Max} + F_i^{Max} - F_i^{min})$$

$$= UP_1(D_i^{in}(j+2))|_j \tag{7.32}$$

The output loop delay will then have the following form.

$$Max(T_{i+1}^1(j+1))|_j \le Max(D_{i+1}^{out,Max},\ UP_1(D_i^{in}(j+2))|_j) \tag{7.33}$$

As will be shown in the next chapter, obtaining the equivalent output delays by looking into cross sections q and p will help analyze a more complex circuit. From the definition of equivalent output delay, we have

$$D_{i,q}^{out}(j+1) = D_a^{qp}(j+1) + F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) \tag{7.34}$$

$$D_{i,p}^{out}(j+1) = D_a^{pq}(j+1) + F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) \tag{7.35}$$

For the initial condition case, $D_a^{qp}(1) = Max(0,\ D_p^{in}(2) - D_q^{in}(2))$ and $D_{i+1}^{24}(1) = 0$, we have

$$D_{i,q}^{out}(1) = Max(F_i(1) + B_i(1),$$

$$D_p^{in}(2) - D_q^{in}(2) + F_i(1) + B_i(1)$$

$$\Rightarrow Max(D_{i,q}^{out}(1))|_{j=0} = Max(F_i^{Max} + B_i^{Max},$$

$$D_p^{in,Max} - D_q^{in,min} + F_i^{Max} + B_i^{Max}) \tag{7.36}$$

From (7.34), we have

$$D_{i,q}^{out}(j+2) = D_a^{qp}(j+2) + F_i(j+2) + D_{i+1}^{24}(j+2) + B_i(j+2) \tag{7.37}$$

By applying the Equal loop–delay theorem to C–element C_{i+1}, we have

$$D_{i+1}^{24}(j+2)$$

$$= Max(0,$$

$$D_{i+1}^{out}(j+1) - B_i(j+1) - D_a^{qp}(j+2) - F_i(j+2) - logic\ delay)$$

$$\Rightarrow D_a^{qp}(j+2) + D_{i+1}^{24}(j+2) \le Max(D_a^{qp}(j+2),$$

$$D_{i+1}^{out}(j+1) - B_i(j+1) - F_i(j+2)) \tag{7.38}$$

From (7.25) and (7.30), we know

$$D_a^{qp}(j+2) \le Max(0,$$
$$D_p^{in}(j+2) - DF_i(j+1)) \tag{7.39}$$
$$\Rightarrow Max(D_a^{qp}(j+2))|_j \le Max(0,$$
$$D_p^{in,Max} - DF_i^{min}) \tag{7.40}$$

From (7.37), (7.38) and (7.39), we have

$$D_{i,q}^{out}(j+2) \le Max(F_i(j+2) + B_i(j+2),$$
$$D_p^{in}(j+2) - F_i(j+1) + F_i(j+2) - B_i(j+1) + B_i(j+2),$$
$$D_{i+1}^{out}(j+1) - B_i(j+1) + B_i(j+2))$$
$$\Rightarrow Max(D_{i,q}^{out}(j+2))|_j \le Max(F_i^{Max} + B_i^{Max},$$
$$D_p^{in,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$
$$D_{i+1}^{out,Max} + B_i^{Max} - B_i^{min}) \tag{7.41}$$

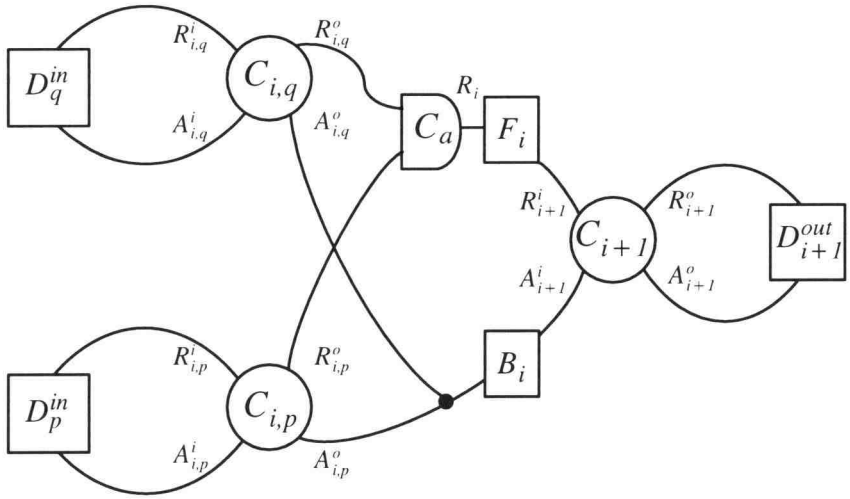
Combining (7.36) and (7.41), we conclude

$$Max(D_{i,q}^{out}(j+1))|_j \le Max(F_i^{Max} + B_i^{Max},$$
$$D_p^{in,Max} - D_q^{in,min} + F_i^{Max} + B_i^{Max},$$
$$D_p^{in,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$
$$D_{i+1}^{out,Max} + B_i^{Max} - B_i^{min}) \tag{7.42}$$

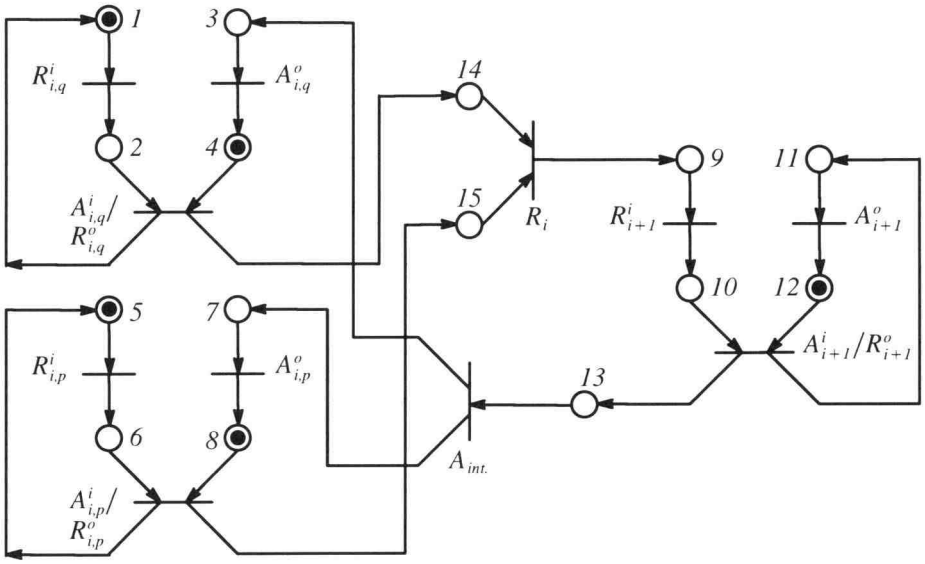
Similarly, we have

$$Max(D_{i,p}^{out}(j+1))|_j \le Max(F_i^{Max} + B_i^{Max},$$
$$D_q^{in,Max} - D_p^{in,min} + F_i^{Max} + B_i^{Max},$$
$$D_q^{in,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$
$$D_{i+1}^{out,Max} + B_i^{Max} - B_i^{min}) \tag{7.43}$$

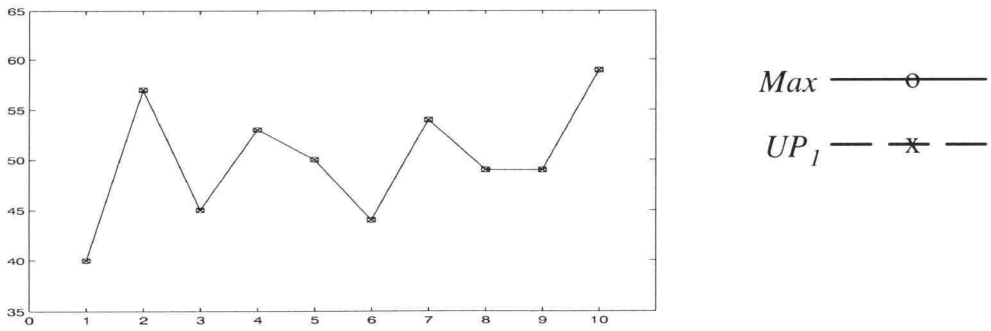
Figure 7.4(b) is a Petri net representation of Figure 7.4(a). To compare the output loop delay result (UP_1) based on our approach (7.33) with the exact upper bound (Max)



(a)



(b)



(c)

Figure 7.4: A Join pipeline and its Petri net model.
(a) General representation of a Join pipeline.
(b) Petri net model.
(c) Result comparison.

obtained using CTSE, several simulations are done and tabulated in Table 7.2. The graphical presentation is shown in Figure 7.4(c). Again, the C–element's physical delay is assumed to be zero. The simulations show that our approach has the same values as the exact bound. However, a general conclusion of our approach leading to the identical result as the exact bound cannot be drawn unless a formal proof is made.

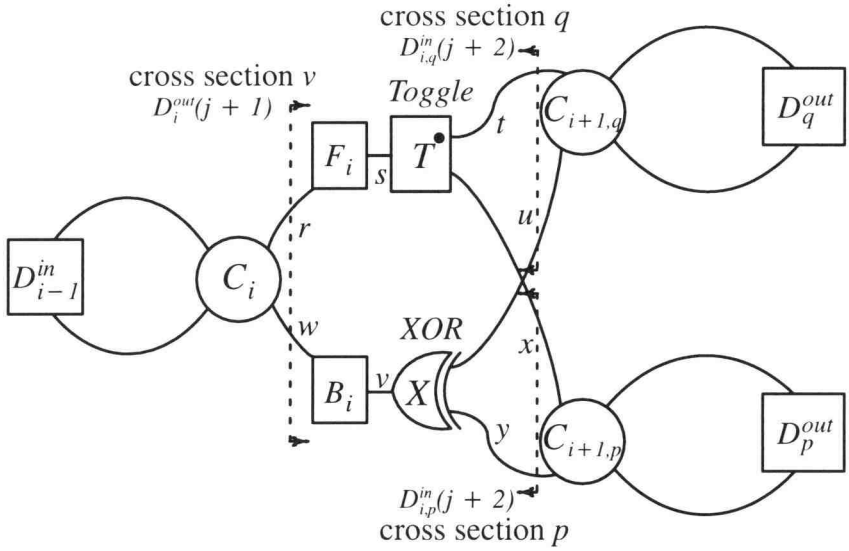
Table 7.2: Comparison of our approximations with the exact bounds using a general Join pipeline as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_q^{in}	[17 25]	[27 47]	[17 20]	[17 20]	[11 21]	[21 31]	[21 51]	[1 11]	[1 11]	[10 21]
D_p^{in}	[2 15]	[12 35]	[2 15]	[12 25]	[4 31]	[4 20]	[4 40]	[4 40]	[4 10]	[4 40]
F_i	[5 20]	[15 25]	[5 15]	[15 25]	[6 25]	[12 25]	[22 25]	[22 25]	[22 25]	[12 25]
B_i	[13 18]	[13 18]	[13 18]	[13 28]	[8 14]	[8 14]	[8 24]	[8 24]	[8 24]	[8 24]
D_{i+1}^{out}	[29 35]	[29 35]	[29 45]	[29 35]	[10 40]	[10 44]	[10 44]	[10 44]	[10 34]	[10 59]
Max	40	57	45	53	50	44	54	49	49	59
UP_1	40	57	45	53	50	44	54	49	49	59

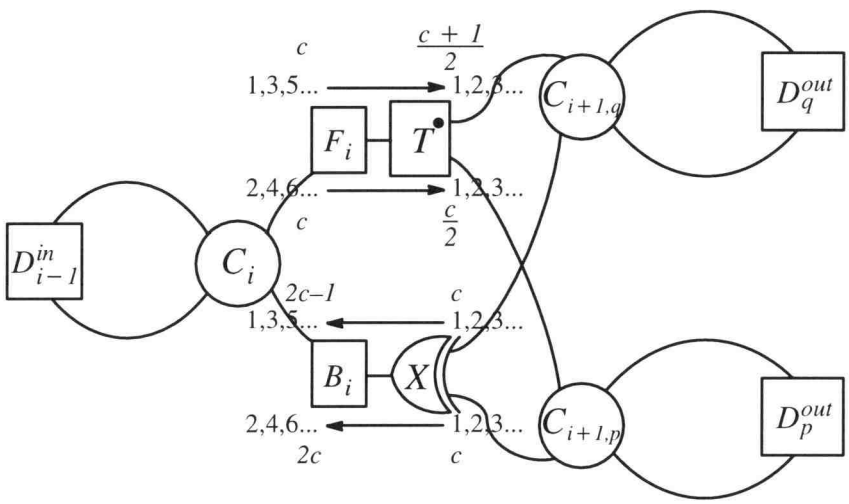
7.3 Performance Of Asynchronous Pipelines With Toggle/Xor Pair

In previous sections, the performance of two dimensional pipelines (Fork and Join) constructed using Event AND (C–element) module have been discussed. In this section, a different two dimensional pipeline built with Toggle and XOR (Event OR) modules is demonstrated and its performance is derived. Figure 7.5(a) depicts a general form of pipelines with Toggle and XOR modules.

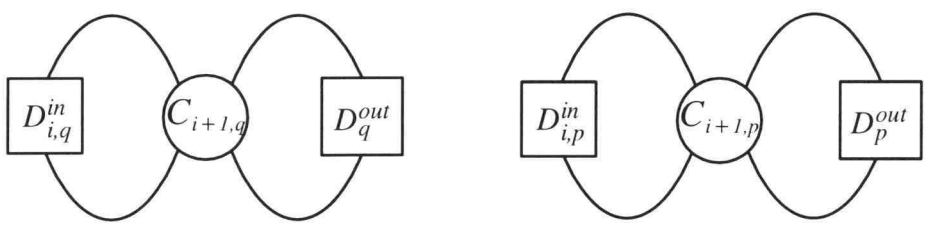
Several approaches to acquiring the performance for a XOR/Toggle pipeline are possible. It appears that using the approach of equivalent input delay by looking into cross sections p and q in Figure 7.5(a) is the most convenient and familiar way and the



(a)



(b)



(c)

Figure 7.5: A two–dimensional micropipeline — Toggle/XOR.
(a) General representation of pipelines with Toggle/XOR.
(b) Token index transformation.
(c) Equivalent circuit.

resulting equivalent circuit is shown in Figure 7.5(c). Due to the function of Toggle, the token index transformation, as shown in Figure 7.5(b), is required when tokens toggle in two different paths. The token index c ($c \geq 1$) is different from j ($j \geq 0$) which we used to adopt. Token index c represents a datum sequence which indicates that the datum is being, or is about to be, processed. For example, the *(j+2)th* datum for F_i, conventionally denoted as $F_i(j+2)$, is also represented as $F_i(c)$, where $c=j+2$. With understanding of Toggle's operation (the first output token goes to the dotted output terminal), the token index transformation becomes obvious and is self–explanatory as shown in Figure 7.5(b).

In turn, we would like to apply the Equal loop–delay theorem to C–elements C_i, $C_{i+1,q}$, and $C_{i+1,p}$. Note that there is no logic delay term for Toggle and XOR modules. They have only propagation delays. Assuming that the propagation delays for Toggle and XOR are constant and equal to D_T and D_X, respectively. By applying the Equal loop–delay theorem to C_i with token travels through the loop $r - s - t - u - v - w - r$, as indicated in Figure 7.5(a), we have

$$D_i^{42}(j_e + 2) = Max(0,$$

$$D_{i-1}^{in}(j_e + 2) - [F_i(j_e + 1) + D_T + D_{i+1,q}^{24}(\frac{(j_e + 1) + 1}{2}) +$$

$$D_X + B_i(2 \times \frac{(j_e + 1) + 1}{2} - 1)])$$

$$= Max(0,$$

$$D_{i-1}^{in}(j_e + 2) - [F_i(j_e + 1) + D_T + D_{i+1,q}^{24}(\frac{j_e}{2} + 1) +$$

$$D_X + B_i(j_e + 1)])$$

where $j_e = 0, 2, 4, 6...$, or

$$D_i^{42}(even) = D_i^{42}(2j + 2)$$

$$= Max(0,$$

$$D_{i-1}^{in}(2j + 2) - [F_i(2j + 1) + D_T + D_{i+1,q}^{24}(j + 1) +$$

$$D_X + B_i(2j + 1)]) \tag{7.44}$$

where $j_e = 2j$ and $j = 0, 1, 2, 3, 4...$

Using similar approach to C_i with token travels through the loop $r - s - x - y - v - w - r$, as indicated in Figure 7.5(a), we have

$$D_i^{42}(j_o + 2) = Max(0,$$

$$D_{i-1}^{in}(j_o + 2) - [F_i(j_o + 1) + D_T + D_{i+1,p}^{24}(\frac{j_o + 1}{2}) +$$

$$D_X + B_i(j_o + 1)])$$

where $j_o = 1, 3, 5, 7...$, or

$$D_i^{42}(odd) = D_i^{42}(2j + 3)$$

$$= Max(0,$$

$$D_{i-1}^{in}(2j + 3) - [F_i(2j + 2) + D_T + D_{i+1,p}^{24}(j + 1) +$$

$$D_X + B_i(2j + 2)]) \tag{7.45}$$

where $j_o = 2j + 1$ and $j = 0, 1, 2, 3, 4...$

Caution should be taken when $C_{i+1,q}$ is considered. Due to Toggle's operation, the token will not travel through the loop $u - v - w - r - s - t$. In stead, it will follow the path of $u - v - w - r - s - x - y - v - w - r - s - t$. Similarly, the token will traverse the loop of $y - v - w - r - s - t - u - v - w - r - s - x$ when $C_{i+1,p}$ is considered. The resulting logic delays are as shown below.

$$D_{i+1,q}^{24}(j + 2) = Max(0,$$

$$D_q^{out}(j + 1) - [D_X + B_i(2j + 1) + D_i^{42}(2j + 2) +$$

$$F_i(2j + 2) + D_T + D_{i+1,p}^{24}(j + 1) + D_X + B_i(2j + 2) +$$

$$D_i^{42}(2j + 3) + F_i(2j + 3) + D_T]) \tag{7.46}$$

$$D_{i+1,p}^{24}(j + 2) = Max(0,$$

$$D_p^{out}(j + 1) - [D_X + B_i(2j + 2) + D_i^{42}(2j + 3) +$$

$$F_i(2j + 3) + D_T + D^{24}_{i+1,q}(j + 2) + D_X + B_i(2j + 3) +$$

$$D^{42}_i(2j + 4) + F_i(2j + 4) + D_T]) \quad (7.47)$$

where $j = 0, 1, 2, 3, 4...$

Now, we are ready to find the delay bounds for $D^{in}_{i,q}(j + 2)$ and $D^{in}_{i,p}(j + 2)$. From previous discussion, we have

$$D^{in}_{i,q}(j + 2) = D_X + B_i(2j + 1) + D^{42}_i(2j + 2) + F_i(2j + 2) + D_T + D^{24}_{i+1,p}(j + 1)$$

$$+ D_X + B_i(2j + 2) + D^{42}_i(2j + 3) + F_i(2j + 3) + D_T \quad (7.48)$$

$$\Rightarrow D^{in}_{i,q}(j + 2) \geq B_i(2j + 1) + B_i(2j + 2) + F_i(2j + 2) + F_i(2j + 3)$$

$$+ 2(D_T + D_X) \quad (7.49)$$

From (7.46), we have

$$D^{24}_{i+1,q}(j + 1) = Max(0,$$

$$D^{out}_q(j) - [D_X + B_i(2j - 1) + D^{42}_i(2j) + F_i(2j) +$$

$$D_T + D^{24}_{i+1,p}(j) + D_X + B_i(2j) +$$

$$D^{42}_i(2j + 1) + F_i(2j + 1) + D_T]) \quad (7.50)$$

From (7.47), we have

$$D^{24}_{i+1,p}(j + 1) = Max(0,$$

$$D^{out}_p(j) - [D_X + B_i(2j) + D^{42}_i(2j + 1) + F_i(2j + 1) +$$

$$D_T + D^{24}_{i+1,q}(j + 1) + D_X + B_i(2j + 1) +$$

$$D^{42}_i(2j + 2) + F_i(2j + 2) + D_T]) \quad (7.51)$$

It should be noted that the terms $D^{42}_i(2j + 2)$, $D^{42}_i(2j + 3)$ and $D^{24}_{i+1,p}(j + 1)$ in (7.48) are not independent. Therefore, if we directly substitute expressions (7.44), (7.45) and (7.51) into (7.48), we will get a larger upper bound approximation than if we can identify their relationship and find the upper bound of the group. That is,

$$Max(D_i^{42}(2j+2) + D_i^{42}(2j+3) + D_{i+1,p}^{24}(j+1))$$

$$\leq Max(D_i^{42}(2j+2)) + Max(D_i^{42}(2j+3)) + Max(D_{i+1,p}^{24}(j+1)) \quad (7.52)$$

Their relationship can be described as follows. From (7.45), we have

$$D_i^{42}(2j+3)$$

$$= Max(0,$$

$$D_{i-1}^{in}(2j+3) - [F_i(2j+2) + D_T + D_{i+1,p}^{24}(j+1) + D_X + B_i(2j+2)])$$

$$\Rightarrow D_i^{42}(2j+3) + D_{i+1,p}^{24}(j+1)$$

$$= Max(D_{i+1,p}^{24}(j+1),$$

$$D_{i-1}^{in}(2j+3) - [F_i(2j+2) + D_T + D_X + B_i(2j+2)]) \quad (7.53)$$

By substituting $D_{i+1,p}^{24}(j+1)$ in (7.51) into (7.53), we get

$$D_i^{42}(2j+3) + D_{i+1,p}^{24}(j+1)$$

$$= Max(0,$$

$$D_p^{out}(j) - [D_X + B_i(2j) + D_i^{42}(2j+1) + F_i(2j+1) + D_T + D_{i+1,q}^{24}(j+1)$$

$$+ D_X + B_i(2j+1) + D_i^{42}(2j+2) + F_i(2j+2) + D_T],$$

$$D_{i-1}^{in}(2j+3) - [F_i(2j+2) + D_T + D_X + B_i(2j+2)])$$

$$\Rightarrow D_i^{42}(2j+3) + D_{i+1,p}^{24}(j+1) + D_i^{42}(2j+2)$$

$$= Max(D_i^{42}(2j+2),$$

$$D_p^{out}(j) - [D_X + B_i(2j) + D_i^{42}(2j+1) + F_i(2j+1) + D_T + D_{i+1,q}^{24}(j+1)$$

$$+ D_X + B_i(2j+1) + F_i(2j+2) + D_T],$$

$$D_{i-1}^{in}(2j+3) + D_i^{42}(2j+2) - [F_i(2j+2) + D_T + D_X + B_i(2j+2)])$$

By substituting $D_i^{42}(2j+2)$ in (7.44) into the above equation and combining with (7.48), we have

$$
\begin{aligned}
&D_{i,q}^{in}(j+2)\\
&= Max(F_i(2j+2) + F_i(2j+3) + B_i(2j+1) + B_i(2j+2) + 2(D_T + D_X),\\
&\qquad D_{i-1}^{in}(2j+2) - F_i(2j+1) + F_i(2j+2) + F_i(2j+3) + B_i(2j+2) + D_T\\
&\qquad\qquad + D_X - \textit{logic delays},\\
&\qquad D_p^{out}(j) - F_i(2j+1) + F_i(2j+3) - B_i(2j) + B_i(2j+2) - \textit{logic delays},\\
&\qquad D_{i-1}^{in}(2j+3) + F_i(2j+3) + B_i(2j+1) + D_T + D_X,\\
&\qquad D_{i-1}^{in}(2j+2) + D_{i-1}^{in}(2j+3) - F_i(2j+1) + F_i(2j+3) - \textit{logic delays})
\end{aligned}
\tag{7.54}
$$

Note that although $B_i(2j)$ and $D_p^{out}(j)$ are not defined when $j=0$ (they appear in the $D_{i+1,p}^{24}(j+1)$), they will not affect the upper bound representation because when $D_{i+1,p}^{24}(1) = 0$, the representation of $D_{i,q}^{in}(j+2)$ is a subset of expression (7.54).

Using a similar approach, we have

$$
\begin{aligned}
D_{i,p}^{in}(j+2) &= D_X + B_i(2j+2) + D_i^{42}(2j+3) + F_i(2j+3) + D_T + D_{i+1,q}^{24}(j+2)\\
&\quad + D_X + B_i(2j+3) + D_i^{42}(2j+4) + F_i(2j+4) + D_T
\end{aligned}
\tag{7.55}
$$

$$
\begin{aligned}
\Rightarrow D_{i,p}^{in}(j+2) &\geq B_i(2j+2) + B_i(2j+3) + F_i(2j+3) + F_i(2j+4)\\
&\quad + 2(D_T + D_X)
\end{aligned}
\tag{7.56}
$$

$$
\begin{aligned}
&D_{i,p}^{in}(j+2)\\
&= Max(F_i(2j+3) + F_i(2j+4) + B_i(2j+2) + B_i(2j+3) + 2(D_T + D_X),\\
&\qquad D_{i-1}^{in}(2j+3) - F_i(2j+2) + F_i(2j+3) + F_i(2j+4) + B_i(2j+3) + D_T\\
&\qquad\qquad + D_X - \textit{logic delays},\\
&\qquad D_q^{out}(j+1) - F_i(2j+2) + F_i(2j+4) - B_i(2j+1) + B_i(2j+3)\\
&\qquad\qquad - \textit{logic delays},\\
&\qquad D_{i-1}^{in}(2j+4) + F_i(2j+4) + B_i(2j+2) + D_T + D_X,
\end{aligned}
$$

$$D^{in}_{i-1}(2j+3) + D^{in}_{i-1}(2j+4) - F_i(2j+2) + F_i(2j+4) - \textit{logic delays})$$

(7.57)

By looking into cross section v, there are two possible token travelling paths. If the token follows the upper path (q path), we have

$$D^{out}_i(j+1) = F_i(2j+1) + D_T + D^{24}_{i+1,q}(j+1) + D_X + B_i(2j+1) \tag{7.58}$$

From (7.50), we have

$$\begin{aligned} D^{out}_i(j+1) = Max(& F_i(2j+1) + B_i(2j+1) + D_T + D_X, \\ & D^{out}_q(j) - F_i(2j) - B_i(2j-1) - B_i(2j) + B_i(2j+1) - D_X \\ & - D_T - \textit{logic delays}) \end{aligned} \tag{7.59}$$

On the other hand, if the token travels the lower path (p path), we have

$$D^{out}_i(j+1) = F_i(2j+2) + D_T + D^{24}_{i+1,p}(j+1) + D_X + B_i(2j+2) \tag{7.60}$$

From (7.51), we have

$$\begin{aligned} D^{out}_i(j+1) = Max(& F_i(2j+2) + B_i(2j+2) + D_T + D_X, \\ & D^{out}_p(j) - F_i(2j+1) - B_i(2j) - B_i(2j+1) + B_i(2j+2) - D_X \\ & - D_T - \textit{logic delays}) \end{aligned} \tag{7.61}$$

If we let $D^{in,min}_{i-1} \le D^{in}_{i-1}(j+2) \le D^{in,Max}_{i-1}$,

$F^{min}_i \le F_i(j+1) \le F^{Max}_i$,

$B^{min}_i \le B_i(j+1) \le B^{Max}_i$,

$D^{out,min}_q \le D^{out}_q(j+1) \le D^{out,Max}_q$, and

$D^{out,min}_p \le D^{out}_p(j+1) \le D^{out,Max}_p$,

then from (7.49), we have

$$min(D^{in}_{i,q}(j+2))|_j \ge 2B^{min}_i + 2F^{min}_i + 2(D_T + D_X) \tag{7.62}$$

From (7.54), we have

$$\begin{aligned} & Max(D^{in}_{i,q}(j+2))|_j \\ & \le Max(2F^{Max}_i + 2B^{Max}_i + 2(D_T + D_X), \end{aligned}$$

$$D_{i-1}^{in,Max} + 2F_i^{Max} - F_i^{min} + B_i^{Max} + D_T + D_X,$$

$$D_p^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$2D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \tag{7.63}$$

From (7.56), we have

$$min(D_{i,p}^{in}(j+2))|_j \geq 2B_i^{min} + 2F_i^{min} + 2(D_T + D_X) \tag{7.64}$$

From (7.57), we have

$$Max(D_{i,p}^{in}(j+2))|_j$$

$$\leq Max(2F_i^{Max} + 2B_i^{Max} + 2(D_T + D_X),$$

$$D_{i-1}^{in,Max} + 2F_i^{Max} - F_i^{min} + B_i^{Max} + D_T + D_X,$$

$$D_q^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$2D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \tag{7.65}$$

The maximum value of $D_i^{out}(j+1)$ will be the maximum of (7.59) and (7.61). That is,

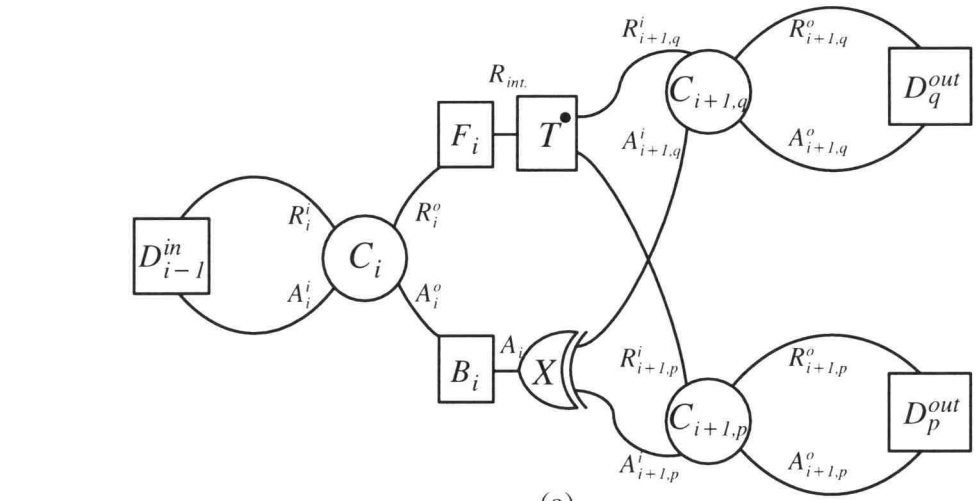
$$Max(D_i^{out}(j+1))|_j$$

$$\leq Max(F_i^{Max} + B_i^{Max} + D_T + D_X),$$

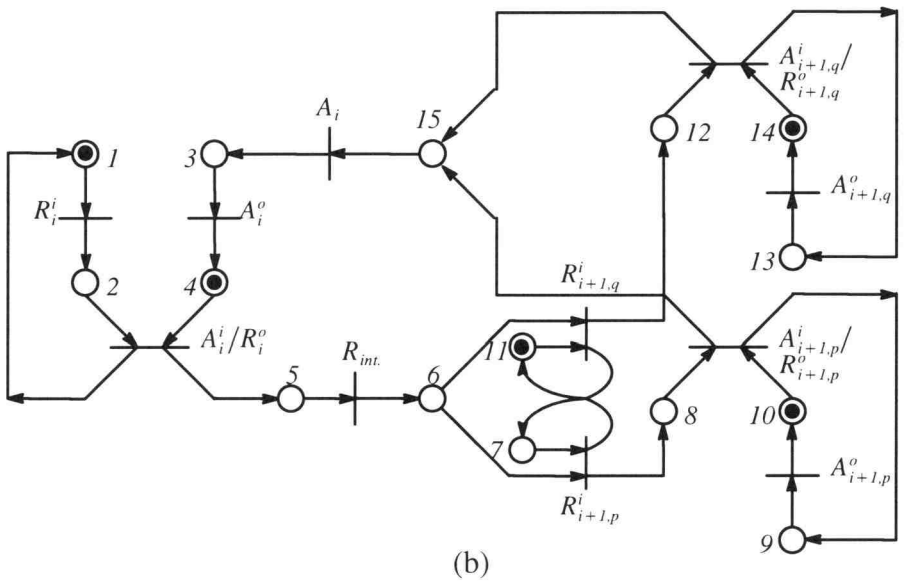
$$D_q^{out,Max} - F_i^{min} + B_i^{Max} - 2B_i^{min} - D_T - D_X,$$

$$D_p^{out,Max} - F_i^{min} + B_i^{Max} - 2B_i^{min} - D_T - D_X) \tag{7.66}$$

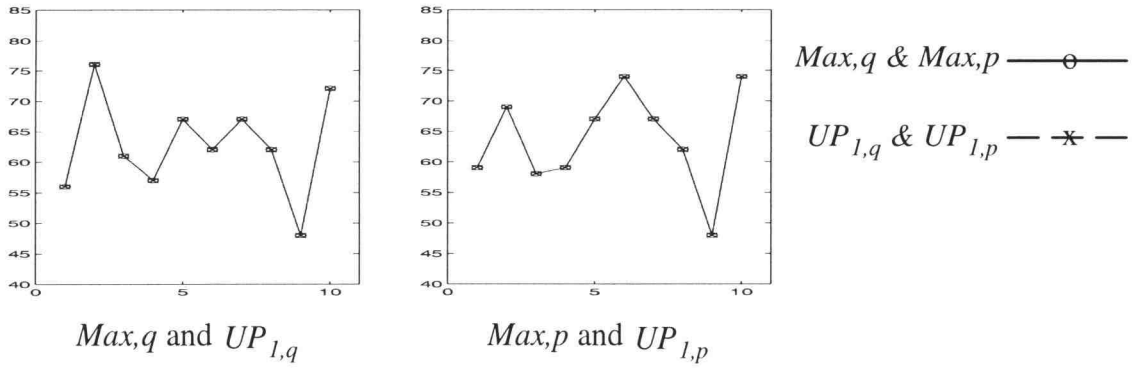
To compare the output loop delays ($UP_{1,q}$ and $UP_{1,p}$) based on our approach with the exact upper bounds (Max,q and Max,p) obtained using CTSE, Figure 7.6(a) and Figure 7.6(b) depict the circuit and Petri net model of Toggle/XOR, respectively. Comparison results are tabulated in Table 7.3. Based upon this table, the graphic result is also plotted in Figure 7.6(c). Note that the physical delays of Toggle and XOR are assumed to be zero in our simulations.



(a)



(b)



(c)

Figure 7.6: A Toggle/XOR pipeline and its Petri net model.
(a) General representation of a Toggle/XOR pipeline.
(b) Petri net model.
(c) Result comparison.

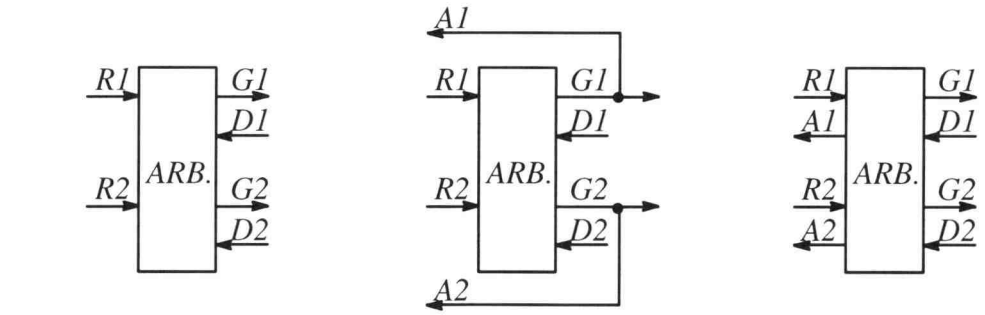
Table 7.3: Comparison of our approximations with the exact bounds using a general Toggle/XOR pipeline as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_{i-1}^{in}	[2 15]	[2 10]	[2 7]	[2 20]	[2 30]	[2 18]	[2 30]	[2 27]	[2 10]	[2 20]
F_i	[1 10]	[8 9]	[8 9]	[1 10]	[2 9]	[6 18]	[6 13]	[6 13]	[8 9]	[6 20]
B_i	[3 18]	[9 15]	[9 20]	[3 18]	[13 20]	[12 12]	[10 15]	[10 15]	[9 15]	[10 15]
D_q^{out}	[29 35]	[29 35]	[29 35]	[29 35]	[19 25]	[19 62]	[9 12]	[9 35]	[29 35]	[9 55]
D_p^{out}	[21 29]	[21 69]	[21 49]	[21 29]	[10 19]	[10 19]	[10 45]	[10 40]	[21 29]	[10 53]
Max,q	56	76	61	57	67	62	67	62	48	72
$UP_{1,q}$	56	76	61	57	67	62	67	62	48	72
Max,p	59	69	58	59	67	74	67	62	48	74
$UP_{1,p}$	59	69	58	59	67	74	67	62	48	74

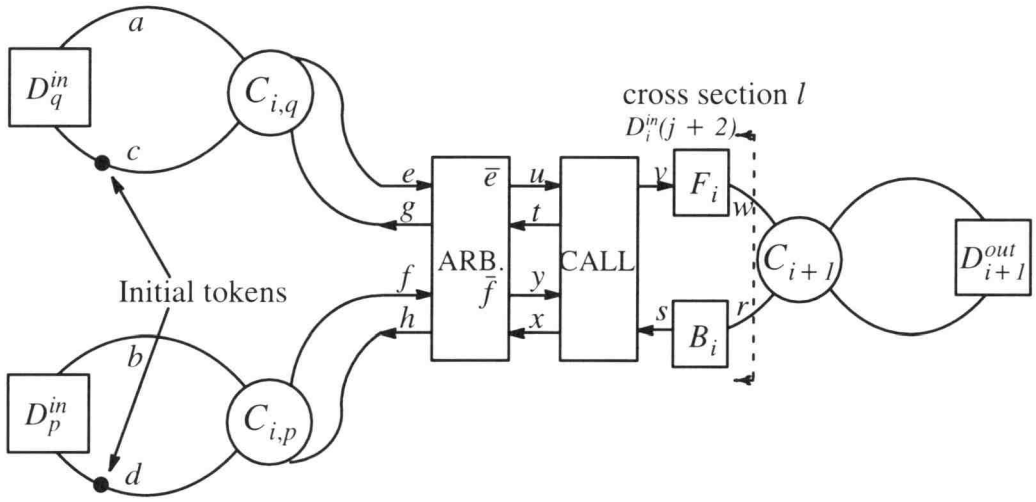
7.4 Performance Of Asynchronous Pipelines With Arbiter/Call Pair

Frequently, a circuit is needed that allows two different parties to compete for one resource. When using asynchronous pipelines, this type of circuit can be implemented with Arbiter and Call modules, as shown in Figure 7.7(b). The performance for an asynchronous pipeline with Arbiter/Call pair is investigated in this section.

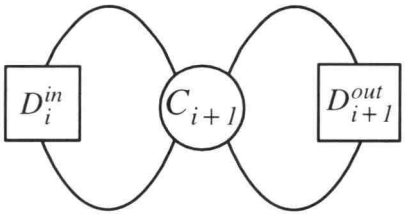
The Arbiter, introduced by Sutherland [9], is shown at the far left graph in Figure 7.7(a). There are no Acknowledge terminals, *A1* and *A2,* attached to Arbiter at all. Depending on the applications, the Acknowledge terminals can be either from Grant terminals, G*1* and *G2,* or Done terminals, *D1* and *D2.* In our performance analysis the Acknowledge signals, *A1* and *A2,* are assumed to be connected with G*1* and *G2* . This is shown on the middle graph in Figure 7.7(a). To avoid the confusion of wiring cross-over, the appearance of Arbiter is redrawn as shown at the far right graph in Figure 7.7(a). Assume that the propagation delays for Arbiter and Call are constant and equal to D_A and D_C, respectively. Note that the mutually exclusive mechanism in Arbi-



(a)



(b)



(c)

Figure 7.7: A two–dimensional micropipeline — Arbiter/Call.
(a) Arbiter representations.
(b) General representation of a pipeline with Arbiter/Call.
(c) Equivalent circuit.

ter is assumed to be very close to the output ends and the time for this mechanism to resolve the competition is very short, such that it is negligible. As a result, the time for one of the Done terminals (*D1* and *D2*) to release the suspended token (if any) is assumed to be zero. If $\bar{e}$ and $\bar{f}$ in Figure 7.7(b) are points right before the mutually exclusive mechanism, the time for a token to travel from e to $\bar{e}$ and f to $\bar{f}$ is a physical delay and equal to D_A. On the other hand, the time for a token to be released at $\bar{e}$ (or $\bar{f}$) and to arrive at u (or y) is a waiting delay. This waiting delay is similar to the waiting or logic delay of a C–element as defined before, but in more general sense.

As long as the delay bound of $D_i^{in}(j+2)$, equivalent input delay by looking into cross section l as shown in Figure 7.7(b), is obtained, the performance of an asynchronous pipeline with Arbiter/Call pair can be derived. There are four possible token paths composing the delay of $D_i^{in}(j+2)$. They are $r-s-t-u-v-w$, $r-s-x-y-v-w$, $r-s-t-y-v-w$ and $r-s-x-u-v-w$. The only difference in terms of delay among these four paths is the delay from t to u, x to y, t to y, and x to u, denoted as $D_i^{tu}(j+1)$, $D_i^{xy}(j+1)$, $D_i^{ty}(j+1)$ and $D_i^{xu}(j+1)$, respectively. All these delays are the time for a resource to wait for a party to grant the permission. Since Arbiter is based on the first come and first serve rule, the $D_i^{in}(j+2)$ is equal to the minimum of these four delays. That is,

from Figure 7.7(b), we have

$$\begin{aligned} D_i^{in}(\bar{j}+2) = min(&B_i(\bar{j}+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(\bar{j}+2), \\ &B_i(\bar{j}+1) + D_C + D_i^{xy}(j'+1) + D_C + F_i(\bar{j}+2), \\ &B_i(\bar{j}+1) + D_C + D_i^{ty}(j'+1) + D_C + F_i(\bar{j}+2), \\ &B_i(\bar{j}+1) + D_C + D_i^{xu}(j'+1) + D_C + F_i(\bar{j}+2)) \qquad (7.67) \end{aligned}$$

$$\geq B_i(\bar{j}+1) + D_C + D_C + F_i(\bar{j}+2) \qquad (7.68)$$

By applying the Equal loop–delay theorem to C–elements $C_{i,q}$ and $C_{i,p}$ in Figure 7.7(b), we have

$$D_{i,q}^{42}(j+2) = Max(0,$$

$$D_q^{in}(j+2) - [D_A + D_i^{\bar{e}u}(j+1)]) \tag{7.69}$$

$$D_{i,p}^{42}(j+2) = Max(0,$$

$$D_p^{in}(j+2) - [D_A + D_i^{\bar{f}y}(j+1)]) \tag{7.70}$$

The token traveling of Arbiter/Call is unlike those of FIFO, Fork, Join and Toggle/XOR. For FIFO, only one token traveling path is possible. As to Fork and Join, there are two token traveling paths and a token will be split into two parts, with one traveling in each path. The token traveling of Toggle/XOR is similar to that of Fork and Join except that the routing path alternates. Therefore, exact expression(s) describing token routing can be derived for each of these modules. Since Arbiter operates on a first come and first serve basis, no exact expression can be possibly deduced to represent its token traveling path . However, we only need to obtain the expression describing the maximum delay in order to find the maximum bound. The notation $Max(f(\cdot))|_p$ denotes the maximum value of $f(\cdot)$ function with respect to all of the possible token traveling paths, denoted as p. This is different from the notation of $Max(f(\cdot))|_j$, which refers to the maximum value of $f(\cdot)$ with respect to j. The most complete representation for the case we are interested in is $Max(f(\cdot))|_{j,p}$, which indicates the maximum value of $f(\cdot)$ with respect to both index j and token traveling path p. Since the token traveling path for FIFO, Fork, Join and Toggle/XOR is known or describable (e.g., the expressions corresponding to all possible paths for Toggle/XOR is explicitly enumerated), the subscript p is ignored. Therefore, $Max(f(\cdot))|_{j,p} = Max(f(\cdot))|_j$ in these cases. Also note that, in general, $Max(f(\cdot))|_{j,p} = Max(Max(f(\cdot))|_j)|_p = Max(Max(f(\cdot))|_p)|_j$.

The delay of $D_i^{ty}(j' + 1)$ is considered first. $D_i^{ty}(j' + 1)$ is the time difference between two tokens arriving at point t and $\bar{f}$. As far as the $D_i^{ty}(j' + 1)$ is concerned, the bottom part of Arbiter can be modeled as a C–element with $\bar{f}$, t, h and y corresponding to R_{i-1}^v, A_i^o, A_{i-1}^v and R_i^o in Figure 3.4(b). If a token arrives at t earlier than at $\bar{f}$ for $j' = k$, then $D_i^{ty}(k + 1) = 0$. If not, $D_i^{ty}(k + 1) > 0$. Therefore, to have maximum value of $D_i^{ty}(j' + 1)$, the path delay to $\bar{f}$ should be as large as possible (longest path to $\bar{f}$) and the path delay to t should be as small as possible (shortest path to t). Considering the case of $Max(D_i^{ty}(1))|_p$, we know the token path to $\bar{f}$ has two different routes. One is $d - b - f - \bar{f}$ and this corresponds to the consideration of the initial condition. The other route is $h - f - \bar{f}$ and this indicates that the bottom part of Arbiter (*G2* in Figure 7.7(a)) has at least granted permission once. We discuss these two routes separately.

Route 1 ($d - b - f - \bar{f}$): The other matching token path to t is $c - a - e - \bar{e} - N(u - v - w - r - s - t)$, where N, representing the number of times a token travels around the upper loop, is a positive integer greater than or equal to *1*. Of course, an infinite number of routings is possible due to the infinite looping. Therefore, to make $D_i^{ty}(1)$ as large as possible among these paths, a route for a token traveling the loop once ($N = 1$) is chosen (note that the less time the token travels on the upper section to arrive at t, the more time this token needs to wait for the incoming token on the lower section to arrive at $\bar{f}$, i.e., $D_i^{ty}(1)$ is larger). That is, based upon the functions of Arbiter and Call, and the initial condition, we have

$$Max(D_i^{ty}(1))|_{p,\ route\ 1}$$

$$= Max(0,$$

$$D_p^{in}(1) + D_{i,p}^{24}(1) + D_A - [D_q^{in}(1) + D_{i,q}^{24}(1) + D_A + D_i^{\bar{e}u}(1) + D_C$$

$$+ F_i(1) + D_{i+1}^{24}(1) + B_i(1) + D_C])$$

$= Max(0,$

$$D_p^{in}(1) - [D_q^{in}(1) + D_i^{\bar{e}u}(1) + D_C + F_i(1) + B_i(1) + D_C]) \quad (7.71)$$

Route 2 ($h - f - \bar{f}$): The other matching token path to t which has the shortest path delay is $y - v - w - r - s - x - u - v - w - r - s - t$. This implies that a token must at least travel the bottom section (i.e., a token appearing on *G2*) once. We should consider the timing right after a token at $\bar{f}$ is granted. The path $y - v - w - r - s - x$ determines how much time the token has been traveling in advance in the $h - f - \bar{f}$ path before the token on $\bar{e}$ grants the permission and travels through the path $u - v - w - r - s - t$. That is, $Max(D_i^{ty}(1))|_{p,\ route\ 2}$

$= Max(0,$

$$D_{i,p}^{42}(j+2) + D_A - [D_C + F_i(j+1) + D_{i+1}^{24}(j+1) + B_i(j+1) + D_C + D_i^{xu}(1) + D_C + F_i(j+2) + D_{i+1}^{24}(j+2) + B_i(j+2) + D_C])$$

Substitute $D_{i,p}^{42}(j+2)$ in (7.70) into above equation, we have

$Max(D_i^{ty}(1))|_{p,\ route\ 2}$

$= Max(0,$

$$D_A - 4D_C - F_i(j+1) - F_i(j+2) - B_i(j+1) - B_i(j+2) - D_{i+1}^{24}(j+1) - D_i^{xu}(1) - D_{i+1}^{24}(j+2),$$

$$D_p^{in}(j+2) - 4D_C - F_i(j+1) - F_i(j+2) - B_i(j+1) - B_i(j+2) - D_{i+1}^{24}(j+1) - D_i^{xu}(1) - D_{i+1}^{24}(j+2) - D_i^{\bar{f}y}(j+1) \quad (7.72)$$

A similar concept used in deriving $Max(D_i^{ty}(1))|_{p,\ route\ 2}$ can be applied to find $Max(D_i^{ty}(j'+2))|_p$. Therefore, the representation of $Max(D_i^{ty}(j'+2))|_p$ would be

$Max(D_i^{ty}(j'+2))|_p$

$= Max(0,$

$$D^{42}_{i,p}(j'' + 2) + D_A - [D_C + F_i(j + 1) + D^{24}_{i+1}(j + 1)$$
$$+ B_i(j + 1) + D_C + D^{xu}_i(j''' + 1) + D_C$$
$$+ F_i(j + 2) + D^{24}_{i+1}(j + 2) + B_i(j + 2) + D_C])$$

Substitute $D^{42}_{i,p}(j'' + 2)$ in (7.70) into the above equation, we have

$$Max(D^{ty}_i(j' + 2))|_p$$
$$= Max(0,$$
$$D_A - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1)$$
$$- B_i(j + 2) - D^{24}_{i+1}(j + 1) - D^{24}_{i+1}(j + 2) - D^{xu}_i(j''' + 1),$$
$$D^{in}_p(j'' + 2) - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1) - B_i(j + 2)$$
$$- D^{24}_{i+1}(j + 1) - D^{24}_{i+1}(j + 2) - D^{xu}_i(j''' + 1) - D^{\bar{t}y}_i(j'' + 1)) \quad (7.73)$$

Similarly, due to the symmetry of Arbiter, from (7.71), (7.72) and (7.73) we have

$$Max(D^{xu}_i(1))|_{p,\ route\ 1}$$
$$= Max(0,$$
$$D^{in}_q(1) - [D^{in}_p(1) + D^{\bar{t}y}_i(1) + D_C + F_i(1) + B_i(1) + D_C]) \quad (7.74)$$

$$Max(D^{xu}_i(1))|_{p,\ route\ 2}$$
$$= Max(0,$$
$$D_A - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1) - B_i(j + 2)$$
$$- D^{24}_{i+1}(j + 1) - D^{ty}_i(1) - D^{24}_{i+1}(j + 2),$$
$$D^{in}_q(j + 2) - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1) - B_i(j + 2)$$
$$- D^{24}_{i+1}(j + 1) - D^{ty}_i(1) - D^{24}_{i+1}(j + 2) - D^{\bar{e}u}_i(j + 1) \quad (7.75)$$

$$Max(D^{xu}_i(j' + 2))|_p$$
$$= Max(0,$$
$$D_A - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1) - B_i(j + 2)$$
$$- D^{24}_{i+1}(j + 1) - D^{24}_{i+1}(j + 2) - D^{ty}_i(j''' + 1),$$

$$D_q^{in}(j'' + 2) - 4D_C - F_i(j + 1) - F_i(j + 2) - B_i(j + 1) - B_i(j + 2)$$
$$- D_{i+1}^{24}(j + 1) - D_{i+1}^{24}(j + 2) - D_i^{ty}(j''' + 1) - D_i^{\bar{e}u}(j'' + 1)) \quad (7.76)$$

Now, we would like to consider the delay representation of $D_i^{tu}(j' + 1)$. This is straightforward, since the token traveling paths corresponding to maximum delay of $D_i^{tu}(j' + 1)$ will be in the upper section only. They are $g - e - \bar{e}$ and $u - v - w - r - s - t$. That is,

$$Max(D_i^{tu}(j' + 1))|_p$$
$$= Max(0,$$
$$D_{i,q}^{42}(j'' + 2) + D_A - [D_C + F_i(j + 1) + D_{i+1}^{24}(j + 1) + B_i(j + 1) + D_C])$$

Substitute $D_{i,q}^{42}(j'' + 2)$ in (7.69) into the above equation and we have

$$Max(D_i^{tu}(j' + 1))|_p$$
$$= Max(0,$$
$$D_A - F_i(j + 1) - B_i(j + 1) - 2D_C - D_{i+1}^{24}(j + 1),$$
$$D_q^{in}(j'' + 2) - F_i(j + 1) - B_i(j + 1) - 2D_C - D_{i+1}^{24}(j + 1)$$
$$- D_i^{\bar{e}u}(j'' + 1)) \quad (7.77)$$

Again, due to the symmetry of Arbiter, from (7.77), we have

$$Max(D_i^{xy}(j' + 1))|_p$$
$$= Max(0,$$
$$D_A - F_i(j + 1) - B_i(j + 1) - 2D_C - D_{i+1}^{24}(j + 1),$$
$$D_p^{in}(j'' + 2) - F_i(j + 1) - B_i(j + 1) - 2D_C - D_{i+1}^{24}(j + 1)$$
$$- D_i^{\bar{f}y}(j'' + 1)) \quad (7.78)$$

where j, j', j'' and j''' are independent and equal to 0, 1, 2, 3 .....

The reason why there are so many different indices (j, j', j'' and j''') is because the token routing paths for this circuit are not fixed, leading to an independent index rela-

tionship. However, the relationship between $\bar{j}$ in (7.67) and j is not independent, as will be shown later.

If we let $D_q^{in,min} \le D_q^{in}(j+2) \le D_q^{in,Max}$,

$$D_p^{in,min} \le D_p^{in}(j+2) \le D_p^{in,Max},$$

$$F_i^{min} \le F_i(j+1) \le F_i^{Max},$$

$$B_i^{min} \le B_i(j+1) \le B_i^{Max},$$

$$D_{i+1}^{out,min} \le D_{i+1}^{out}(j+1) \le D_{i+1}^{out,Max},$$

and $D_q^{in}(1)$ and $D_p^{in}(1)$ represent the initial conditions (assume $D_q^{in,min} \le D_q^{in}(1) \le D_q^{in,Max}$ and $D_p^{in,min} \le D_p^{in}(1) \le D_p^{in,Max}$), we can approximate the upper bound of $D_i^{in}(j+2)$ based on expression (7.67).

$$B_i(\bar{j}+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(\bar{j}+2)$$

$$\le B_i(\bar{j}+1) + D_C + Max(D_i^{tu}(j'+1))|_p + D_C + F_i(\bar{j}+2)$$

According to expression (7.77) and the routing path, we have $\bar{j} = j$. The above expression can be rewritten as

$$B_i(\bar{j}+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(\bar{j}+2)$$

$$= Max(B_i(\bar{j}+1) + D_C + D_C + F_i(\bar{j}+2),$$

$$D_A + F_i(j+2) - F_i(j+1) - D_{i+1}^{24}(j+1),$$

$$D_q^{in}(j''+2) + F_i(j+2) - F_i(j+1) - D_{i+1}^{24}(j+1) - D_i^{\bar{e}u}(j''+1))$$

$$\Rightarrow Max(B_i(\bar{j}+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(\bar{j}+2))|_j$$

$$\le Max(F_i^{Max} + B_i^{Max} + 2D_C,$$

$$F_i^{Max} - F_i^{min} + D_A,$$

$$D_q^{in,Max} + F_i^{Max} - F_i^{min}) \qquad (7.79)$$

Similarly, we have

$$Max(B_i(\bar{j}+1) + D_C + D_i^{xy}(j'+1) + D_C + F_i(\bar{j}+2))|_j$$

$$\leq Max(F_i^{Max} + B_i^{Max} + 2D_C,$$
$$F_i^{Max} - F_i^{min} + D_A,$$
$$D_p^{in,Max} + F_i^{Max} - F_i^{min}) \quad (7.80)$$

To find the maximum delay of $B_i(\bar{j} + 1) + D_C + D_i^{ty}(j' + 1) + D_C + F_i(\bar{j} + 2)$, where $j = 0, 1, 2, 3$ ....., a two–step approach is adopted. First, we find the maximum delay based on $Max(D_i^{ty}(1))|_{p,\ route\ 1}$ and $Max(D_i^{ty}(1))|_{p,\ route\ 2}$. From (7.71) and the routing path, we have $\bar{j} = 0$.

$$B_i(1) + D_C + D_i^{ty}(1) + D_C + F_i(2)$$
$$\leq B_i(1) + D_C + Max(D_i^{ty}(1))|_{p,\ route\ 1} + D_C + F_i(2)$$
$$\leq Max(F_i(2) + B_i(1) + 2D_C,$$
$$D_p^{in}(1) + F_i(2) - F_i(1) - D_q^{in}(1) - D_{i,q}^{24}(1) - D_i^{\bar{e}u}(1)) \quad (7.81)$$

From (7.72) and the corresponding routing path, we have $\bar{j} = j + 1$, That is,

$$B_i(\bar{j} + 1) + D_C + D_i^{ty}(1) + D_C + F_i(\bar{j} + 2)$$
$$= B_i(j + 2) + D_C + D_i^{ty}(1) + D_C + F_i(j + 3)$$
$$\leq B_i(j + 2) + D_C + Max(D_i^{ty}(1))|_{p,\ route\ 2} + D_C + F_i(j + 3)$$
$$\leq Max(F_i(j + 3) + B_i(j + 2) + 2D_C,$$
$$F_i(j + 3) - F_i(j + 1) - F_i(j + 2) - B_i(j + 1)$$
$$+ D_A - 2D_C - D_{i+1}^{24}(j + 1) - D_i^{xu}(1) - D_{i+1}^{24}(j + 2),$$
$$D_p^{in}(j + 2) - F_i(j + 1) - F_i(j + 2) + F_i(j + 3) - B_i(j + 1) - 2D_C$$
$$- D_{i+1}^{24}(j + 1) - D_i^{xu}(1) - D_{i+1}^{24}(j + 2) - D_i^{\bar{f}y}(j + 1) \quad (7.82)$$

Second, we find the maximum delay based on $Max(D_i^{ty}(j' + 2))|_p$. Note that this is equivalent to finding $B_i(\bar{j} + 1) + D_C + D_i^{ty}(j' + 2) + D_C + F_i(\bar{j} + 2)$. From expression (7.73) and the corresponding routing path, we know $\bar{j} = j + 1$. That is,

$$
\begin{aligned}
& B_i(\bar{j}+1) + D_C + D_i^{ty}(j'+2) + D_C + F_i(\bar{j}+2) \\
&= B_i(j+2) + D_C + D_i^{ty}(j'+2) + D_C + F_i(j+3) \\
&\le B_i(j+2) + D_C + Max(D_i^{ty}(j'+2))|_p + D_C + F_i(j+3) \\
&\le Max(F_i(j+3) + B_i(j+2) + 2D_C, \\
&\qquad F_i(j+3) - F_i(j+1) - F_i(j+2) - B_i(j+1) + D_A - 2D_C \\
&\qquad\quad - D_{i+1}^{24}(j+1) - D_{i+1}^{24}(j+2) - D_i^{xu}(j'''+1), \\
&\qquad D_p^{in}(j''+2) + F_i(j+3) - F_i(j+1) - F_i(j+2) - B_i(j+1) \\
&\qquad\quad - 2D_C - D_{i+1}^{24}(j+1) - D_{i+1}^{24}(j+2) \\
&\qquad\quad - D_i^{xu}(j'''+1) - D_i^{\bar{t}y}(j''+1)) \qquad (7.83)
\end{aligned}
$$

Combining (7.81), (7.82) and (7.83), we have

$$
\begin{aligned}
& Max(B_i(\bar{j}+1) + D_C + D_i^{ty}(j'+1) + D_C + F_i(\bar{j}+2))|_j \\
&\le Max(F_i^{Max} + B_i^{Max} + 2D_C, \\
&\qquad D_p^{in,Max} + F_i^{Max} - F_i^{min} - D_q^{in,min}, \\
&\qquad F_i^{Max} - 2F_i^{min} - B_i^{min} + D_A - 2D_C, \\
&\qquad D_p^{in,Max} + F_i^{Max} - 2F_i^{min} - B_i^{min} - 2D_C) \qquad (7.84)
\end{aligned}
$$

Similarly, we have

$$
\begin{aligned}
& Max(B_i(\bar{j}+1) + D_C + D_i^{xu}(j'+1) + D_C + F_i(\bar{j}+2))|_j \\
&\le Max(F_i^{Max} + B_i^{Max} + 2D_C, \\
&\qquad D_q^{in,Max} + F_i^{Max} - F_i^{min} - D_p^{in,min}, \\
&\qquad F_i^{Max} - 2F_i^{min} - B_i^{min} + D_A - 2D_C, \\
&\qquad D_q^{in,Max} + F_i^{Max} - 2F_i^{min} - B_i^{min} - 2D_C) \qquad (7.85)
\end{aligned}
$$

In conclusion, to find the lower and upper bounds of $D_i^{in}(j+2)$, the following expressions are observed. From (7.67) and (7.68), we have

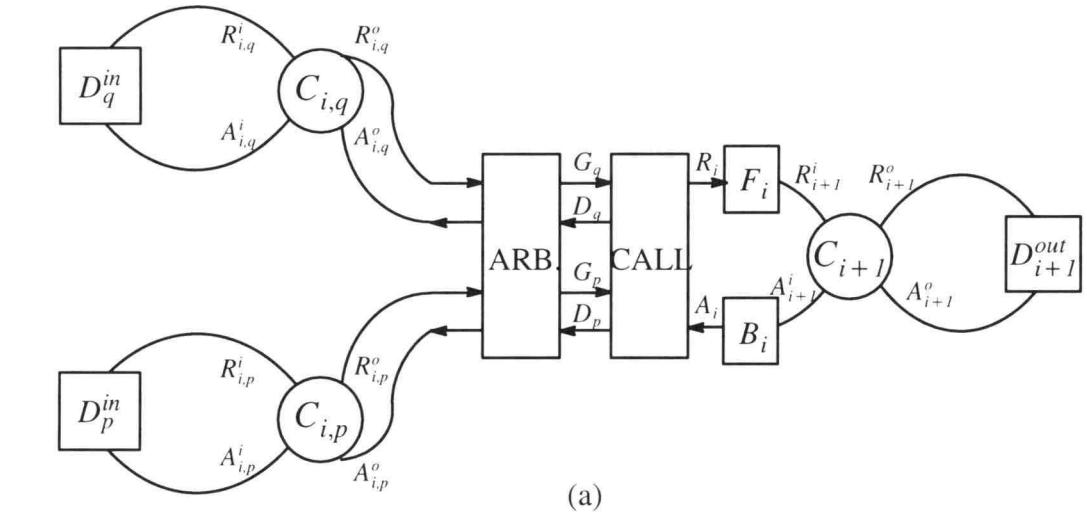
$$min(D_i^{in}(j+2))|_j$$

$$\geq F_i^{min} + B_i^{min} + 2D_C \tag{7.86}$$

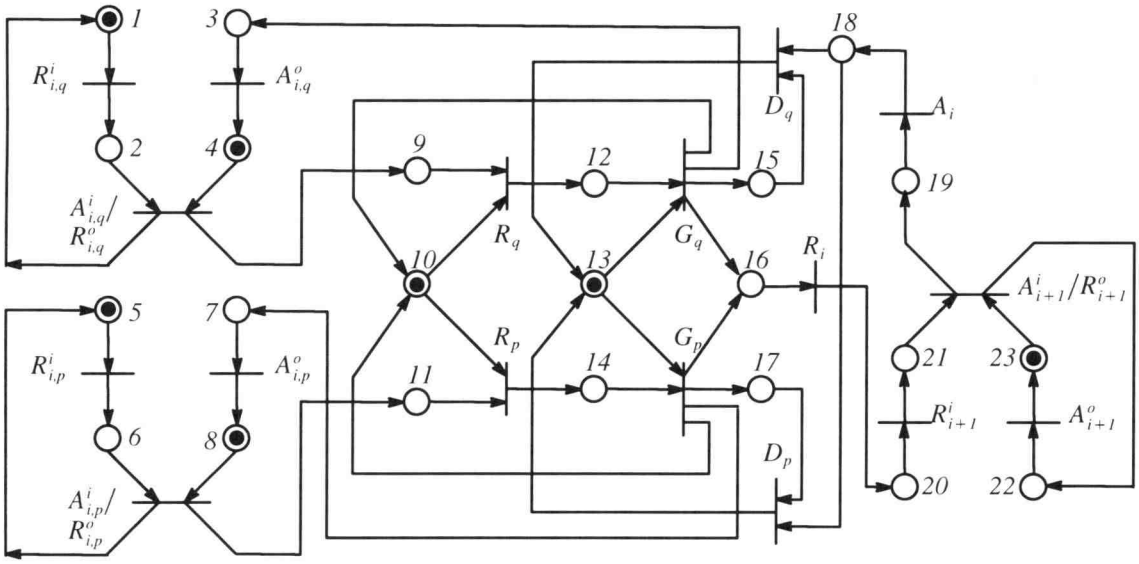
$$\begin{aligned}
&Max(D_i^{in}(j+2))|_j \\
&= Max(min(B_i(j+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(j+2), \\
&\qquad B_i(j+1) + D_C + D_i^{xy}(j'+1) + D_C + F_i(j+2), \\
&\qquad B_i(j+1) + D_C + D_i^{ty}(j'+1) + D_C + F_i(j+2), \\
&\qquad B_i(j+1) + D_C + D_i^{xu}(j'+1) + D_C + F_i(j+2)))|_j \\
&= min(Max(B_i(j+1) + D_C + D_i^{tu}(j'+1) + D_C + F_i(j+2))|_j, \\
&\qquad Max(B_i(j+1) + D_C + D_i^{xy}(j'+1) + D_C + F_i(j+2))|_j, \\
&\qquad Max(B_i(j+1) + D_C + D_i^{ty}(j'+1) + D_C + F_i(j+2))|_j, \\
&\qquad Max(B_i(j+1) + D_C + D_i^{xu}(j'+1) + D_C + F_i(j+2))|_j)
\end{aligned}$$

$$\leq min((7.79),\ (7.80),\ (7.84),\ (7.85)) \tag{7.87}$$

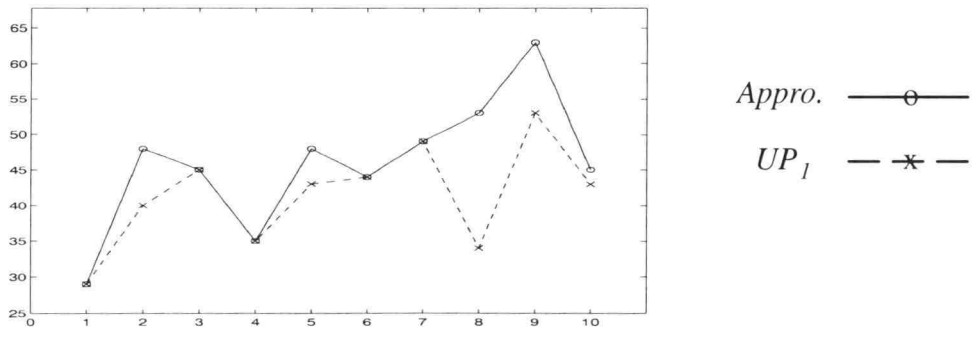
According to the work by Hulgaard and Burns [21], their algorithm and tool cannot identify the exact bounds for a system which contains mutually exclusive component (such as the Arbiter module) and whose performance is data–dependent (such as the Selector module for data selection, as will be shown in the next section). However, they can still approximate the exact upper bound from the right direction, i.e., with the result greater than or equal to the exact upper bound. Figure 7.8(a) shows the circuit under simulation and Figure 7.8(b) demonstrates its corresponding Petri net representation. Table 7.4 compares our approximations (UP_1) with the approximations implemented using CTSE (*Appro.*) for the output loop delay of Arbiter/Call pipeline. The results show that our approximation may approach the exact value better than CTSE. This can also be depicted in Figure 7.8(c). Note that the physical delays of Arbiter and Call are assumed to be zero in this simulation.



(a)



(b)



(c)

Figure 7.8: An Arbiter/Call pipeline and its Petri net model.
(a) General representation of an Arbiter/Call pipeline.
(b) Petri net model.
(c) Result comparison.

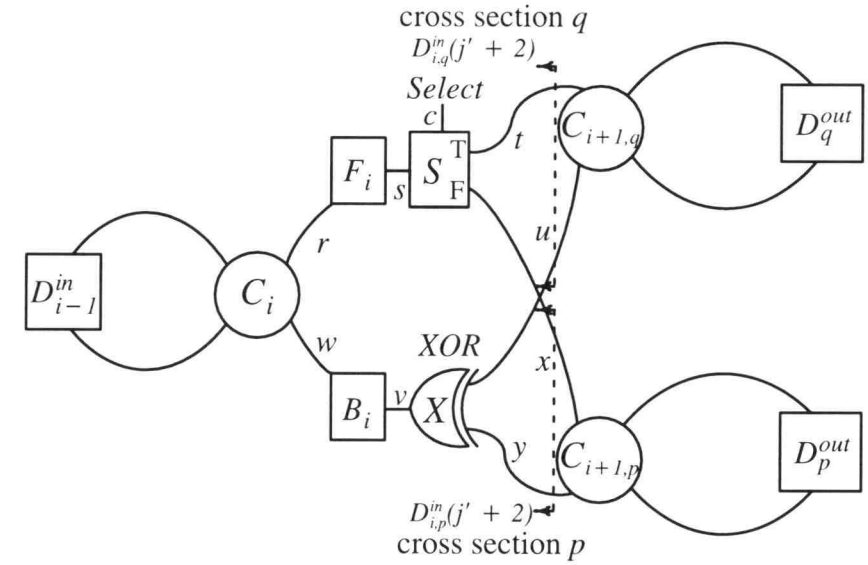
Table 7.4: Comparison of our approximations with the other approximations using a general Arbiter/Call pipeline as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_q^{in}	[10 20]	[10 41]	[17 37]	[27 37]	[20 47]	[21 31]	[21 51]	[10 21]	[10 41]	[27 47]
D_p^{in}	[3 4]	[9 30]	[12 35]	[12 25]	[10 35]	[4 20]	[4 40]	[9 40]	[9 40]	[12 35]
F_i	[12 20]	[12 30]	[20 25]	[20 25]	[12 25]	[12 25]	[22 25]	[2 15]	[2 25]	[15 25]
B_i	[2 4]	[8 10]	[13 20]	[3 10]	[13 18]	[8 14]	[8 24]	[8 14]	[8 14]	[13 18]
D_{i+1}^{out}	[10 29]	[20 24]	[19 35]	[19 30]	[9 20]	[10 44]	[10 44]	[10 34]	[10 24]	[29 35]
Appro.	29	48	45	35	48	44	49	53	63	45
UP_1	29	40	45	35	43	44	49	34	53	43

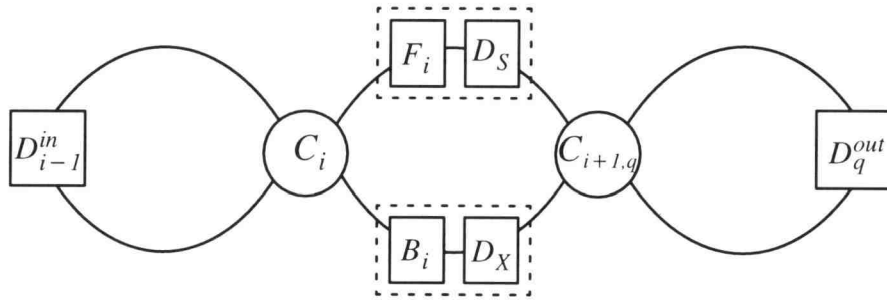
7.5 Performance Of Asynchronous Pipelines With Select/Xor Pair

It is common to have an application that is data–dependent and needs a mechanism to steer the incoming token to an appropriate destination based upon the nature of the data. This mechanism can be implemented using Select/XOR pair in asynchronous pipelines, as shown in Figure 7.9(a). It is this section's purpose to find the performance of asynchronous pipelines with Select/XOR pair.

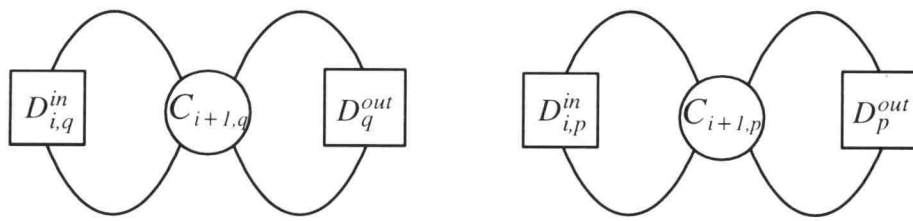
We first define the function of Select and its token sequence on c terminal. When the token value on c is High, the incoming token on s will be steered to the output terminal marked T. On the other hand, if token value on c is Low, the token on s is routed to F terminal. Among the values of token sequence on c, if we let S_T represent the number of consecutive Highs and S_F the number of consecutive Lows, then we have $N_T^m \le S_T \le N_T^M$ and $N_F^m \le S_F \le N_F^M$, where N_T^m is the minimum number of consecutive Highs, N_T^M the maximum number of consecutive Highs, N_F^m the minimum number of consecutive Lows and N_F^M the maximum number of consecutive Lows in a control sequence. For example, consider the following sequence



(a)



(b)



(c)

Figure 7.9: A two–dimensional micropipeline — Select/XOR.
(a) General representation of a pipeline with Select/XOR.
(b) The special case when $N^m_F = 0$.
(c) Equivalent circuit.

c: { HHH L HHH LLLL HH LLLLLLLLL HHHH LL HHHHH }

we have $N_T^m = 2$, $N_T^M = 5$, $N_F^m = 1$ and $N_F^M = 9$, i.e., $2 \le S_T \le 5$ and $1 \le S_F \le 9$. Note that, if $N_T^M = 1$ (implying $N_T^m = 1$) and $N_F^M = 1$ (implying $N_F^m = 1$), this is equivalent to the case of Toggle/XOR pair as discussed before. If $N_F^m = 0$ (implying $N_F^M = 0$), this indicates that no any Lows in the token sequence and the resulting pipeline becomes a linear pipeline as shown in Figure 7.9(b). It is also true if $N_T^m = 0$. The delays D_S and D_X in Figure 7.9(b) are the propagation delays for Select and XOR modules, respectively. Since the above two special cases have been discussed, this section will focus on the case when $N_T^M \ge N_T^m \ge 1$ and $N_F^M \ge N_F^m \ge 1$ (precluding the case of $N_T^M = N_F^M = 1$). Note that, no logic delay or waiting delay occurs in the Select and XOR modules.

We still employ the equivalent delay technique to approach the performance of the Select/XOR pipeline. By looking into cross sections q and p in Figure 7.9(a), its equivalent circuits are shown in Figure 7.9(c). Again, the notations $Max(f(\cdot))|_p$ and $min(f(\cdot))|_p$ denote the maximum and minimum delays of $f(\cdot)$ function with respect to all the possible token traveling paths, respectively.

Now, consider the case when $N_T^M=1$. The shortest token traverse route for $D_{i,q}^{in}(j+2)$ is $u - v - w - r - N_F^m(s - x - y - v - w - r - s) - t$. If $N_T^M \ge 2$, the shortest path becomes $u - v - w - r - s - t$. Therefore, the traverse path can be represented as follows in terms of propagation and logic delays.

$min(D_{i,q}^{in}(j' + 2))|_p$ when $N_T^M=1$

$$
\begin{aligned}
&= D_X + B_i(j+1) + D_i^{42}(j+2) + F_i(j+2) + \\
&\quad [D_S + D_{i+1,p}^{24}(j'' + 1) + D_X + B_i(j+2) + D_i^{42}(j+3) + F_i(j+3) + \\
&\quad D_S + D_{i+1,p}^{24}(j'' + 2) + D_X + B_i(j+3) + D_i^{42}(j+4) + F_i(j+4) + \\
&\quad \cdots\cdots
\end{aligned}
$$

$$D_S + D^{24}_{i+1,p}(j'' + N^m_F) + D_X + B_i(j + N^m_F + 1) + D^{42}_i(j + N^m_F + 2) + F_i(j + N^m_F + 2)] + D_S \quad (7.88)$$

$$\geq \sum_{l=2}^{N^m_F+2} F_i(j + l) + \sum_{l=1}^{N^m_F+1} B_i(j + l) + (N^m_F + 1)D_X + (N^m_F + 1)D_S \quad (7.89)$$

$min(D^{in}_{i,q}(j' + 2))|_p$ when $N^M_T \geq 2$

$$= D_X + B_i(j + 1) + D^{42}_i(j + 2) + F_i(j + 2) + D_S$$

$$\geq F_i(j + 2) + B_i(j + 1) + D_X + D_S \quad (7.90)$$

Similarly, when $N^M_T \geq 1$, $Max(D^{in}_{i,q}(j + 2))|_p$ has the traverse path of $u - v - w - r - N^M_F(s - x - y - v - w - r - s) - t$. It can be represented as follows.

$Max(D^{in}_{i,q}(j' + 2))|_p$

$$= D_X + B_i(j + 1) + D^{42}_i(j + 2) + F_i(j + 2) +$$

$$[D_S + D^{24}_{i+1,p}(j'' + 1) + D_X + B_i(j + 2) + D^{42}_i(j + 3) + F_i(j + 3) +$$

$$D_S + D^{24}_{i+1,p}(j'' + 2) + D_X + B_i(j + 3) + D^{42}_i(j + 4) + F_i(j + 4) +$$

$$\cdots\cdots$$

$$D_S + D^{24}_{i+1,p}(j'' + N^M_F) + D_X + B_i(j + N^M_F + 1) + D^{42}_i(j + N^M_F + 2) + F_i(j + N^M_F + 2)] + D_S \quad (7.91)$$

The representation of $D^{24}_{i+1,p}(j'' + 1)$ depends on the token traveling route. Since the smaller $D^{in}_{i,p}(j' + 2)$ is, the larger $D^{24}_{i+1,p}(j'' + 1)$ is, we need to find the shortest token traveling path of $D^{in}_{i,p}(j' + 2)$ in order to get largest value of $D^{24}_{i+1,p}(j'' + 1)$. Among different routes, the maximum value of $D^{24}_{i+1,p}(j'' + 1)$ occurs when the token travels the upper path (c = High) only once, right before it travels N^M_F times on the lower path (c = Low). That is, the control sequence is { ...LL H LL...LLL(N^M_F times) H... } (assume $N^m_T = 1$). With this kind of control sequence, we have,

$$Max(D^{24}_{i+1,p}(j'' + 1))|_p$$

$$= Max(0,$$

$$D^{out}_p(j'') - [D_X + B_i(j) + D^{42}_i(j + 1) + F_i(j + 1) + D_S + D^{24}_{i+1,q}(j''' + 1)$$

$$+ D_X + B_i(j + 1) + D^{42}_i(j + 2) + F_i(j + 2) + D_S]) \quad (7.92)$$

$$D^{42}_i(j + 2)$$

$$= Max(0,$$

$$D^{in}_{i-1}(j + 2) - [F_i(j + 1) + D_S + D^{24}_{i+1,q}(j'' + 1) + D_X$$

$$+ B_i(j + 1)]) \quad (7.93)$$

The following logic delays become obvious with N^M_F times Low control signal. They are

$$D^{24}_{i+1,p}(j'' + 2)$$

$$= Max(0,$$

$$D^{out}_p(j'' + 1) - [D_X + B_i(j + 2) + D^{42}_i(j + 3) + F_i(j + 3) + D_S]) \quad (7.94)$$

. . . . . .

$$D^{24}_{i+1,p}(j'' + N^M_F)$$

$$= Max(0,$$

$$D^{out}_p(j'' + N^M_F - 1) - [D_X + B_i(j + N^M_F) + D^{42}_i(j + N^M_F + 1)$$

$$+ F_i(j + N^M_F + 1) + D_S]) \quad (7.95)$$

$$D^{42}_i(j + 3)$$

$$= Max(0,$$

$$D^{in}_{i-1}(j + 3) - [F_i(j + 2) + D_S + D^{24}_{i+1,p}(j' + 1) + D_X$$

$$+ B_i(j + 2)]) \quad (7.96)$$

. . . . . .

$$D^{42}_i(j + N^M_F + 2)$$

$$= Max(0,$$

$$D^{in}_{i-1}(j + N^M_F + 2) - [F_i(j + N^M_F + 1) + D_S + D^{24}_{i+1,p}(j' + N^M_F) + D_X$$

$$B_i(j + N^M_F + 1) \tag{7.97}$$

If N^M_F is a large number, the complexity of representation of (7.91) in terms of the expressions ranging from (7.92) to (7.97) grows rapidly. To reduce the complexity, we let $N^M_F = 2$ ($N^m_T = 1$ is implied in deriving (7.92)) in (7.91). That is,

$$\begin{aligned}
&Max(D^{in}_{i,q}(j' + 2))|_p \\
&= D_X + B_i(j + 1) + D^{42}_i(j + 2) + F_i(j + 2) + \\
&\quad [D_S + D^{24}_{i+1,p}(j'' + 1) + D_X + B_i(j + 2) + D^{42}_i(j + 3) + F_i(j + 3) + \\
&\quad D_S + D^{24}_{i+1,p}(j'' + 1) + D_X + B_i(j + 3) + D^{42}_i(j + 4) + F_i(j + 4)] + \\
&\quad D_S
\end{aligned} \tag{7.98}$$

If we let $D^{in,min}_{i-1} \le D^{in}_{i-1}(j + 2) \le D^{in,Max}_{i-1}$,

$F^{min}_i \le F_i(j + 1) \le F^{Max}_i$,

$B^{min}_i \le B_i(j + 1) \le B^{Max}_i$,

$D^{out,min}_p \le D^{out}_p(j + 1) \le D^{out,Max}_p$,

$D^{out,min}_q \le D^{out}_q(j + 1) \le D^{out,Max}_q$,

then according to (7.89) and (7.90), we have

$$\begin{aligned}
&min(D^{in}_{i,q}(j' + 2))|_j \text{ when } N^M_T=1 \\
&\ge min(min(D^{in}_{i,q}(j + 2))|_p)|_j \\
&\ge (N^m_F + 1)F^{min}_i + (N^m_F + 1)B^{min}_i + (N^m_F + 1)D_X + (N^m_F + 1)D_S
\end{aligned} \tag{7.99}$$

$$\begin{aligned}
&min(D^{in}_{i,q}(j' + 2))|_j \text{ when } N^M_T \ge 2 \\
&\ge F^{min}_i + B^{min}_i + D_X + D_S
\end{aligned} \tag{7.100}$$

Similarly, by substituting expressions (7.92) to (7.97) with $N^M_F = 2$ into (7.98) and doing some simple mathematical manipulations, we have

$$
\begin{aligned}
& Max(D_{i,q}^{in}(j+2))|_j \\
& \le Max(Max(D_{i,q}^{in}(j+2))|_p)|_j \\
& \le Max(3F_i^{Max} + 3B_i^{Max} + 3D_X + 3D_S, \\
& \quad D_{i-1}^{in,Max} + 3F_i^{Max} - F_i^{min} + 2B_i^{Max} + 2D_X + 2D_S, \\
& \quad D_p^{out,Max} + 2F_i^{Max} + 2B_i^{Max} + 2D_X + 2D_S, \\
& \quad D_{i-1}^{in,Max} + D_p^{out,Max} + 2F_i^{Max} - 2F_i^{min} + 2B_i^{Max} - 2B_i^{min}, \\
& \quad 2D_{i-1}^{in,Max} + 2F_i^{Max} - F_i^{min} + B_i^{Max} + D_X + D_S, \\
& \quad 2D_p^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min}, \\
& \quad D_{i-1}^{in,Max} + D_p^{out,Max} + F_i^{Max} + 2B_i^{Max} - B_i^{min} + D_X + D_S, \\
& \quad D_p^{out,Max} + D_{i-1}^{in,Max} + 2F_i^{Max} - F_i^{min} + B_i^{Max} + D_X + D_S, \\
& \quad 2D_{i-1}^{in,Max} + D_p^{out,Max} + 2F_i^{Max} - 3F_i^{min} + B_i^{Max} - 2B_i^{min} - D_X - D_S, \\
& \quad 3D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}, \\
& \quad 2D_{i-1}^{in,Max} + D_p^{out,Max} + 2F_i^{Max} - 2F_i^{min}, \\
& \quad 2D_p^{out,Max} + D_{i-1}^{in,Max} + F_i^{Max} - 2F_i^{min} + B_i^{Max} - 2B_i^{min} - D_X - D_S, \\
& \quad 2D_{i-1}^{in,Max} + D_p^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min}, \\
& \quad 2D_{i-1}^{in,Max} + 2D_p^{out,Max} + F_i^{Max} - 3F_i^{min} + B_i^{Max} - 3B_i^{min} - 2D_X - 2D_S, \\
& \quad 3D_{i-1}^{in,Max} + D_p^{out,Max} + F_i^{Max} - 2F_i^{min} - B_i^{min} - D_X - D_S, \\
& \quad 3D_{i-1}^{in,Max} + 2D_p^{out,Max} + F_i^{Max} - 4F_i^{min} - 3B_i^{min} - 3D_X - 3D_S) \qquad (7.101)
\end{aligned}
$$

Since the upper path and lower path in Figure 7.9(a) are symmetrical, we can immediately obtain the upper and lower bounds of $D_{i,p}^{in}(j+2)$ from (7.99) and (7.101). That is,

$$
\begin{aligned}
& min(D_{i,p}^{in}(j+2))|_j \text{ when } N_F^M=1 \\
& \ge (N_T^m + 1)F_i^{min} + (N_T^m + 1)B_i^{min} + (N_T^m + 1)D_X + (N_T^m + 1)D_S \qquad (7.102)
\end{aligned}
$$

$$min(D_{i,p}^{in}(j+2))|_j \text{ when } N_F^M \geq 2$$

$$\geq F_i^{min} + B_i^{min} + D_X + D_S \tag{7.103}$$

$$Max(D_{i,p}^{in}(j+2))|_j$$

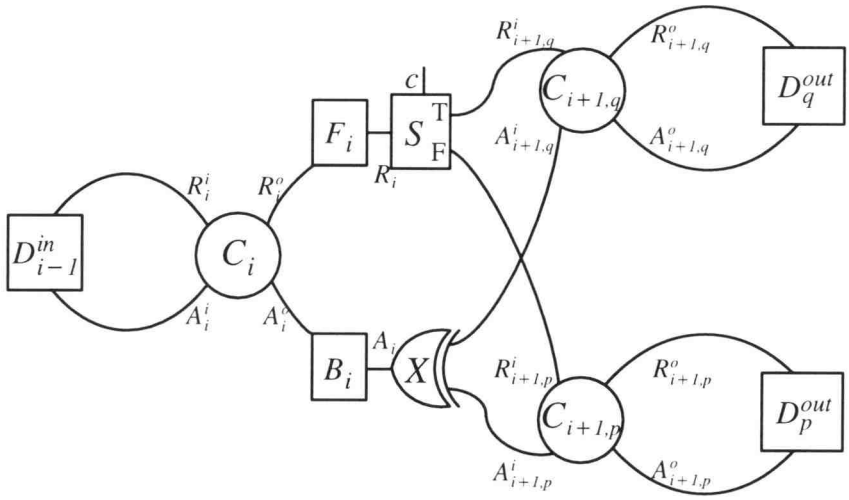
$$\leq \text{expression of (7.101) with the replacement of } D_p^{out,Max} \text{ with } D_q^{out,Max} \tag{7.104}$$

The overall performance of Figure 7.9(a) can be obtained by substituting expressions (7.99) through (7.104) into Figure 7.9(c).

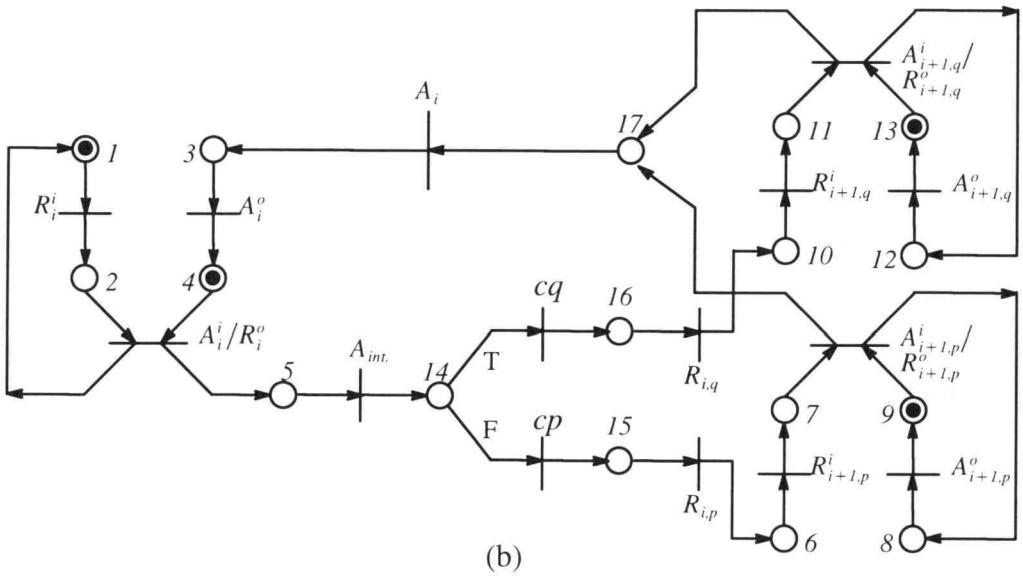
Figure 7.10(b) shows the Petri net model of the general Select/XOR circuit shown in Figure 7.10(a). Unfortunately, CTSE does not allow the user to specify finite values for N_T^M, N_T^m, N_F^M and N_F^m; that is, the tool only simulates the Petri net with $N_T^M = \infty$ and $N_F^M = \infty$ for the upper bound. Therefore, the output loop delays for both upper and lower sections are infinite. Table 7.5 summarizes the simulation results based upon our approach with N_T^M=2, N_T^m=1, N_F^M=2 and N_F^m=1. Note that the physical delays of Select and XOR are assumed to be zero in our simulations. The result is also plotted in Figure 7.10(c).

Table 7.5: The result of our approximation to a general Select/XOR pipeline.

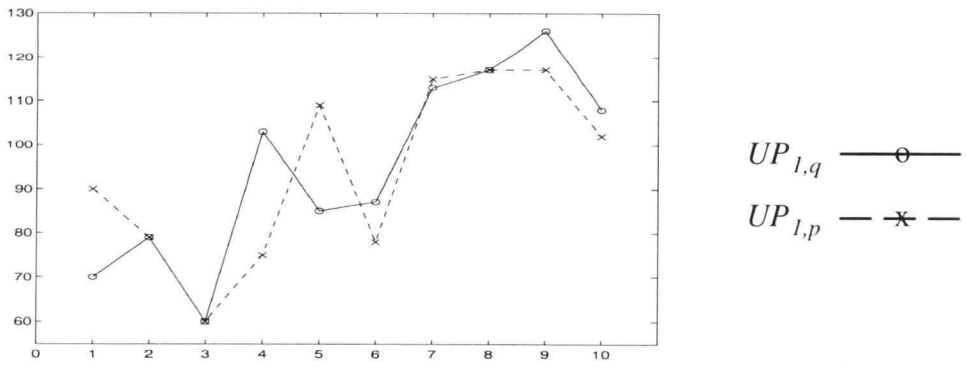
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_{i-1}^{in}	[2 15]	[2 10]	[2 10]	[2 10]	[2 10]	[2 15]	[2 15]	[6 30]	[16 35]	[2 15]
F_i	[1 8]	[1 18]	[9 10]	[9 15]	[9 15]	[11 18]	[1 13]	[10 21]	[10 11]	[1 10]
B_i	[3 8]	[3 8]	[3 10]	[3 10]	[3 10]	[3 8]	[3 10]	[14 17]	[14 17]	[3 8]
D_q^{out}	[19 25]	[9 15]	[5 10]	[9 20]	[29 48]	[19 25]	[30 35]	[10 35]	[10 35]	[25 30]
D_p^{out}	[11 15]	[11 15]	[11 15]	[12 45]	[2 35]	[11 35]	[29 34]	[22 24]	[22 44]	[31 33]
$UP_{1,q}$	70	79	60	103	85	87	113	117	126	108
$UP_{1,p}$	90	79	60	75	109	78	115	117	117	102



(a)



(b)



(c)

Figure 7.10: A Select/XOR pipeline and its Petri net model.
(a) General representation of a Select/XOR pipeline.
(b) Petri net model.
(c) Computation result.

7.6 Summary

In this chapter, based upon the model of the linear pipeline developed in previous chapter, the performance of two–dimensional micropipelines, including Fork, Join, Toggle/XOR, Arbiter/Call and Select/XOR pipelines, is discussed. Several simulations are provided to compare our approximations with the exact bound (except for the cases of Arbiter/Call and Select/XOR pipelines) obtained using CTSE. Based upon these simulations, it shows that our approximations may lead to the exact bounds in certain cases, depending on the set of stage–delays. If the result based on our approach does not match the exact bound, it must be greater than the exact bound. This indicates that, based on the simulation results, our approximation has successfully approached the exact bound from the right hand side. In the Arbiter/Call pipeline, simulations show that our approach has better approximation than CTSE. With certain assumption (N_T^M, N_T^m, N_F^M and N_F^m are finite and known), our methods can also approach the upper bound of Select/XOR pipeline, whose upper bound result can not be directly obtained using the CTSE tool.

8. PERFORMANCE ANALYSIS AND DESIGN EXAMPLES OF TWO–DIMENSIONAL PIPELINES

In Chapter 5, the performance analysis of linear (one–dimensional) pipelines (FIFOs) has been discussed thoroughly. Designing a FIFO that satisfies the delay bound requirement has also been practiced. In Chapter 7, the performance of five two–dimensional pipeline constructs based on the asynchronous modules were analyzed individually. These two–dimensional pipelines include Fork, Join, Toggle/XOR, Arbiter/Call and Select/XOR. It is this chapter's purpose to analyze the performance of larger systems consisting of FIFO, Fork, Join and Toggle/XOR based on previous analyses. The results are compared with the exact values. Therefore, Arbiter/Call and Select/XOR are excluded in this system since their exact bounds cannot be obtained through CTSE. A design example is also implemented.

8.1 A Performance Analysis Example — Open–Loop System

A system which is called open loop and contains FIFO, Fork, Join and Toggle/XOR is provided in this section for the performance analysis. The definition of open–loop system is best understood by demonstrating through an example and will be discussed later in this section. A closed–loop system is also possible and will be described in next section. Again, only the maximum output loop delay (maximum cycle time) is calculated. Figure 8.1 shows the open–loop system to be used here. Our goal is to obtain the maximum value of output loop delays $T^l_{i+2,t}(j+1)$, $T^l_{i+2,u}(j+1)$ and $T^l_{i+2,p}(j+1)$. However, to achieve the goal, several constrains, described as follows, are imposed on the method of approximation developed in the previous chapter.

1. To find $T^l_{i+2,p}(j+1)$, $D^{in}_f(j+2)$ and $D^{in}_b(j+2)$ should be known in advance.
2. To find $D^{in}_b(j+2)$, $D^{out}_c(j+1)$ should be calculated first.

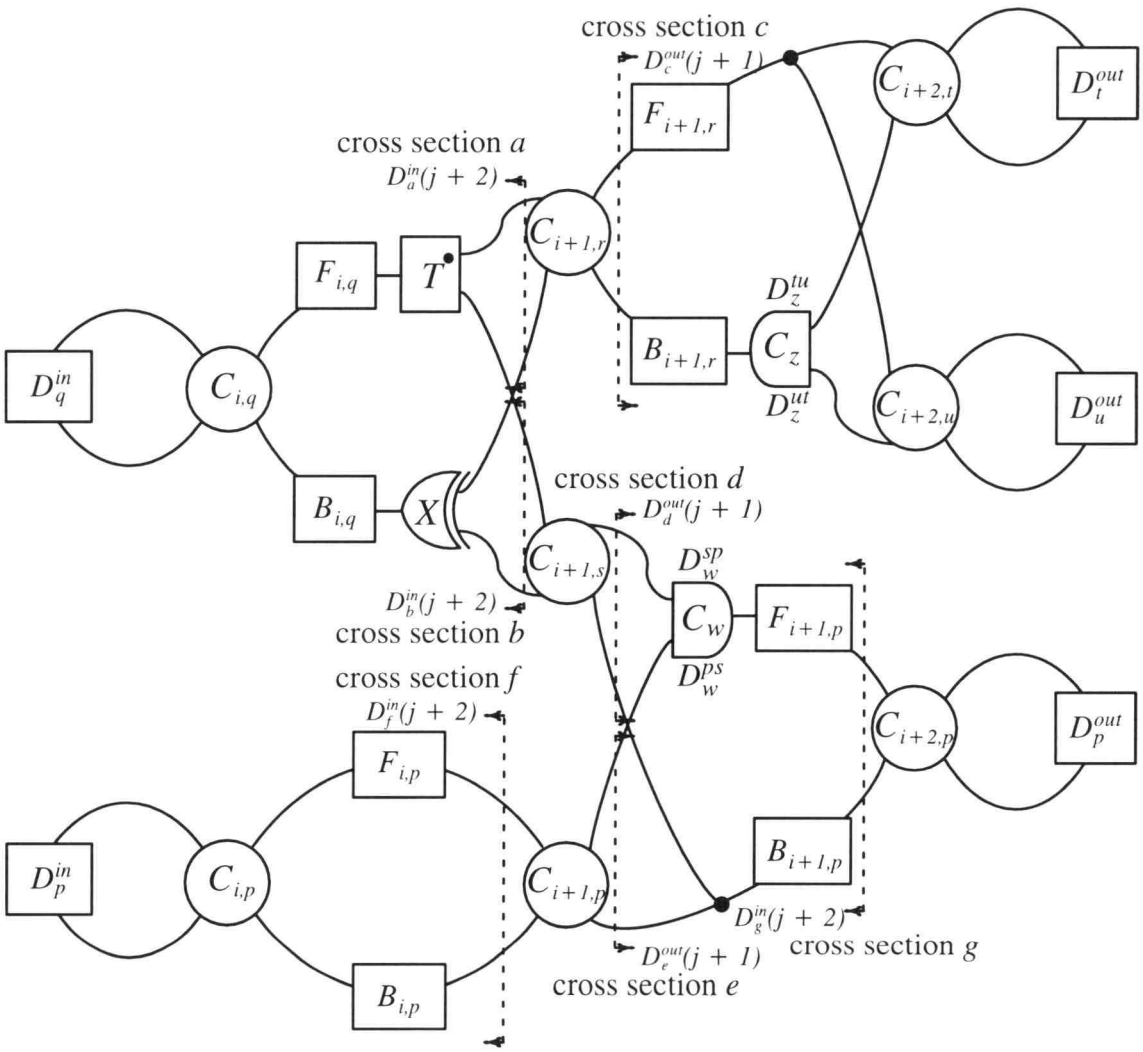


Figure 8.1: A two–dimensional pipeline system — open loop.

3. To obtain $T^{l}_{i+2,t}(j+1)$ and $T^{l}_{i+2,u}(j+1)$, $D^{in}_{a}(j+2)$ should be known first.

4. To get $D^{in}_{a}(j+2)$, $D^{out}_{d}(j+1)$ should be obtained in advance.

5. To get $D^{out}_{d}(j+1)$, $D^{in}_{f}(j+2)$ should be obtained in advance.

6. $D^{in}_{f}(j+2)$ and $D^{out}_{c}(j+1)$ can be obtained directly from given forward and backward delays without requiring the knowledge of any other equivalent input/output delays.

According to the constraints stated above, a flow chart to obtain $Max(T^{l}_{i+2,t}(j+1))|_{j}$, $Max(T^{l}_{i+2,u}(j+1))|_{j}$ and $Max(T^{l}_{i+2,p}(j+1))|_{j}$ is suggested and depicted in Figure 8.2. This flow chart also demonstrates the definition of "open loop." This system is *open loop* since to obtain $Max(T^{l}_{i+2,p}(j+1))|_{j}$, we need to find $D^{out,Max}_{c}$, $D^{in,Max}_{f}$ and $D^{in,Max}_{b}$, three of which are a function of forward and backward delays only. If, for example, $D^{in,Max}_{b}$ is not only a function of forward and backward delays but also the function of $D^{in,Max}_{b}$ itself, then the system will be *closed loop*. The same argument can be applied to obtain $Max(T^{l}_{i+2,t}(j+1))|_{j}$ and $Max(T^{l}_{i+2,u}(j+1))|_{j}$.

Based on the previous chapter's result, the expression corresponding to each equivalent input/output delay in Figure 8.2 is represented as a function of forward and backward delays and described as follows. The physical delays of Toggle (D_T) and XOR (D_X) are assumed to be zero.

1. $$D^{out,Max}_{c} \le Max(F^{Max}_{i+1,r} + B^{Max}_{i+1,r},\ D^{out,Max}_{u} + B^{Max}_{i+1,r} - B^{min}_{i+1,r},\ D^{out,Max}_{t} + B^{Max}_{i+1,r} - B^{min}_{i+1,r}) \quad (8.1)$$

$$D^{in,Max}_{f} = Max(F^{Max}_{i,p} + B^{Max}_{i,p},\ D^{in,Max}_{p} + F^{Max}_{i,p} - F^{min}_{i,p}) \quad (8.2)$$

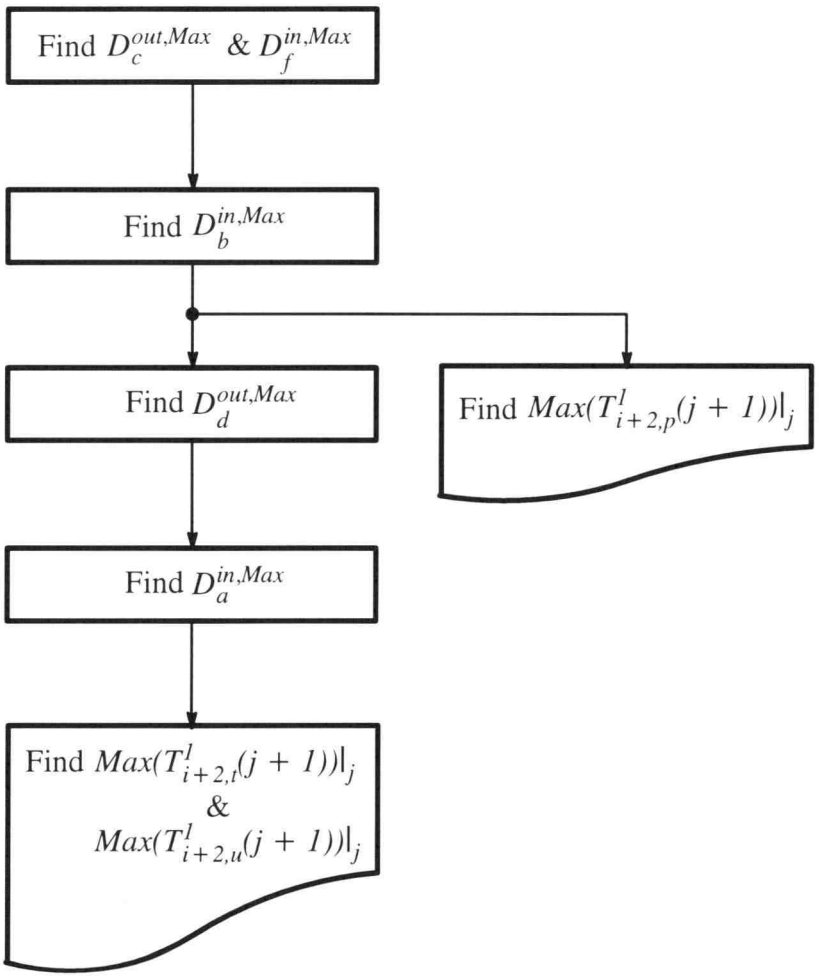


Figure 8.2: The flow chart for obtaining maximum output loop delays of Figure 8.1.

2. $D_b^{in,Max}$

$$\leq Max(\, 2F_{i,q}^{Max} + 2B_{i,q}^{Max} \,,$$

$$D_q^{in,Max} + 2F_{i,q}^{Max} - F_{i,q}^{min} + B_{i,q}^{Max},$$

$$D_c^{out,Max} + F_{i,q}^{Max} - F_{i,q}^{min} + B_{i,q}^{Max} - B_{i,q}^{min},$$

$$2D_q^{in,Max} + F_{i,q}^{Max} - F_{i,q}^{min} \,) \tag{8.3}$$

3. $D_d^{out,Max}$

$$\leq Max(\, F_{i+1,p}^{Max} + B_{i+1,p}^{Max},$$

$$D_p^{in,Max} + F_{i,p}^{Max} - D_q^{in,min} - 2F_{i,q}^{min} - B_{i,q}^{min} + F_{i+1,p}^{Max} + B_{i+1,p}^{Max},$$

$$D_f^{in,Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + B_{i+1,p}^{Max} - B_{i+1,p}^{min},$$

$$D_p^{out,Max} + B_{i+1,p}^{Max} - B_{i+1,p}^{min}) \tag{8.4}$$

$$Max(T_{i+2,p}^{l}(j+1))|_j$$

$$\leq Max(\, D_p^{out,Max},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max},$$

$$D_b^{in,Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min},$$

$$D_f^{in,Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min}) \tag{8.5}$$

4. $D_a^{in,Max}$

$$\leq Max(\, 2F_{i,q}^{Max} + 2B_{i,q}^{Max} \,,$$

$$D_q^{in,Max} + 2F_{i,q}^{Max} - F_{i,q}^{min} + B_{i,q}^{Max},$$

$$D_d^{out,Max} + F_{i,q}^{Max} - F_{i,q}^{min} + B_{i,q}^{Max} - B_{i,q}^{min},$$

$$2D_q^{in,Max} + F_{i,q}^{Max} - F_{i,q}^{min} \,) \tag{8.6}$$

5. $Max(T_{i+2,t}^{l}(j+1))|_j$

$$\leq Max(\, D_t^{out,Max},$$

$$F_{i+1,r}^{Max} + B_{i+1,r}^{Max},$$

$$D_u^{out,Max} + F_{i+1,r}^{Max} - F_{i+1,r}^{min} + B_{i+1,r}^{Max} - B_{i+1,r}^{min},$$

$$D_a^{in,Max} + F_{i+1,r}^{Max} - F_{i+1,r}^{min}) \tag{8.7}$$

$$Max(T_{i+2,u}^{I}(j+1))|_j$$

$$\leq Max(D_u^{out,Max},$$

$$F_{i+1,r}^{Max} + B_{i+1,r}^{Max},$$

$$D_t^{out,Max} + F_{i+1,r}^{Max} - F_{i+1,r}^{min} + B_{i+1,r}^{Max} - B_{i+1,r}^{min},$$

$$D_a^{in,Max} + F_{i+1,r}^{Max} - F_{i+1,r}^{min}) \tag{8.8}$$

The system in Figure 8.1 is also modeled using Petri net and simulated using CTSE. The results based on expressions (8.1) to (8.8) and CTSE are tabulated in Table 8.1. This table shows that most of the results using our approach are actually equal to the exact values. Appendix A demonstrates the process name of CTSE code and corresponding Petri net model for some basic modules used to construct the system of Figure 8.1. The actual CTSE code for each basic module and the CTSE code itself of Figure 8.1 are shown at Appendix B.

Table 8.1: Comparison of our approximations with the exact bounds using Figure 8.1 (open–loop system) as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D_q^{in}	[2 15]	[2 15]	[2 5]	[12 15]	[12 25]	[12 15]	[2 5]	[10 12]	[10 12]	[10 32]
D_p^{in}	[12 15]	[12 15]	[20 25]	[20 25]	[2 10]	[2 10]	[2 30]	[9 18]	[9 15]	[34 45]
$F_{i,q}$	[1 1]	[4 11]	[14 19]	[4 19]	[4 9]	[4 9]	[4 9]	[9 18]	[2 8]	[12 18]
$B_{i,q}$	[13 18]	[13 28]	[3 8]	[3 18]	[3 18]	[3 8]	[3 8]	[4 9]	[4 9]	[4 9]
$F_{i,p}$	[9 10]	[9 10]	[9 10]	[19 30]	[5 10]	[5 10]	[5 20]	[5 12]	[5 9]	[5 11]
$B_{i,p}$	[6 18]	[16 18]	[6 8]	[6 8]	[6 8]	[6 8]	[6 8]	[4 7]	[4 7]	[4 7]
$F_{i+1,r}$	[10 10]	[10 20]	[1 8]	[10 28]	[5 18]	[5 8]	[5 8]	[9 18]	[9 15]	[4 10]
$B_{i+1,r}$	[3 18]	[13 18]	[3 10]	[3 5]	[3 5]	[3 5]	[3 5]	[9 12]	[9 12]	[11 12]
$F_{i+1,p}$	[11 20]	[11 20]	[5 10]	[5 15]	[5 15]	[5 10]	[5 15]	[7 10]	[7 10]	[7 9]
$B_{i+1,p}$	[13 18]	[3 18]	[3 5]	[3 5]	[3 5]	[3 5]	[3 15]	[9 25]	[5 10]	[5 9]
D_t^{out}	[9 35]	[9 25]	[39 55]	[29 45]	[5 10]	[15 60]	[15 40]	[15 30]	[5 10]	[15 20]
D_u^{out}	[20 39]	[20 29]	[11 15]	[11 25]	[31 59]	[31 49]	[31 59]	[12 48]	[12 58]	[22 28]
D_p^{out}	[21 49]	[21 39]	[2 9]	[2 9]	[2 9]	[22 39]	[22 39]	[22 39]	[22 75]	[22 30]
Max,t	59	88	61	98	74	60	80	78	97	76
$UP_{1,t}$	59	88	61	100	74	60	80	78	97	76
Max,u	59	88	69	98	70	65	64	78	97	76
$UP_{1,u}$	59	88	69	100	70	65	80	78	97	76
Max,p	68	87	77	87	91	77	81	68	75	72
$UP_{1,p}$	68	87	77	87	91	77	81	68	75	72

8.2 A Performance Analysis Example — Closed–Loop System

As shown in the previous section, the results for each two–dimensional pipeline module developed in Chapter 7 can be immediately applied to obtain output loop delays of an open–loop system constructed by these modules. Table 8.1 shows that this direct application turns out to have a good approximation. In this section, a similar approach

will be applied to the closed–loop system, as shown in Figure 8.3. The goal for this system is to obtain the maximum output loop delay $Max(T^{l}_{i+3}(j+1))|_j$. The dependency in the process for approaching this goal is depicted in Figure 8.4. Based on this dependency flow chart, it is said that the system in Figure 8.3 is a closed–loop system since $D_f^{in,Max}$, for example, is not only a function of forward and backward delays but also the maximum delay $D_f^{in,Max}$ itself. In terms of token index, $D_f^{in}(j+k)$ is a function of $D_f^{in}(j+l)$ in this closed–loop system, where $k > l \geq 2$. The equivalent input and output delays looking into different cross sections are described as follows.

$$\begin{aligned} Max(T^{l}_{i+3}(j+1))|_j \leq Max(&D_{i+3}^{out,Max}, \\ &F_{i+2}^{Max} + B_{i+2}^{Max}, \\ &D_f^{in,Max} + F_{i+2}^{Max} - F_{i+2}^{min}, \\ &D_g^{in,Max} + F_{i+2}^{Max} - F_{i+2}^{min}) \end{aligned} \tag{8.9}$$

$$\begin{aligned} D_f^{in,Max} \leq Max(&F_{i+1,q}^{Max} + B_{i+1,q}^{Max}, \\ &D_b^{in,Max} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}) \end{aligned} \tag{8.10}$$

$$\begin{aligned} D_b^{in,Max} \leq Max(&F_i^{Max} + B_i^{Max}, \\ &D_e^{out,Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min}, \\ &D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \end{aligned} \tag{8.11}$$

$$\begin{aligned} D_e^{out,Max} \leq Max(&F_{i+1,p}^{Max} + B_{i+1,p}^{Max}, \\ &D_i^{out,Max} + B_{i+1,p}^{Max} - B_{i+1,p}^{min}) \end{aligned} \tag{8.12}$$

$$\begin{aligned} D_i^{out,Max} \leq Max(&F_{i+2}^{Max} + B_{i+2}^{Max}, \\ &F_{i+1,q}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} + B_{i+2}^{Max}, \\ &D_f^{in,Max} + F_{i+2}^{Max} - F_{i+2}^{min} + B_{i+2}^{Max} - B_{i+2}^{min}, \\ &D_{i+3}^{out,Max} + B_{i+2}^{Max} - B_{i+2}^{min}) \end{aligned} \tag{8.13}$$

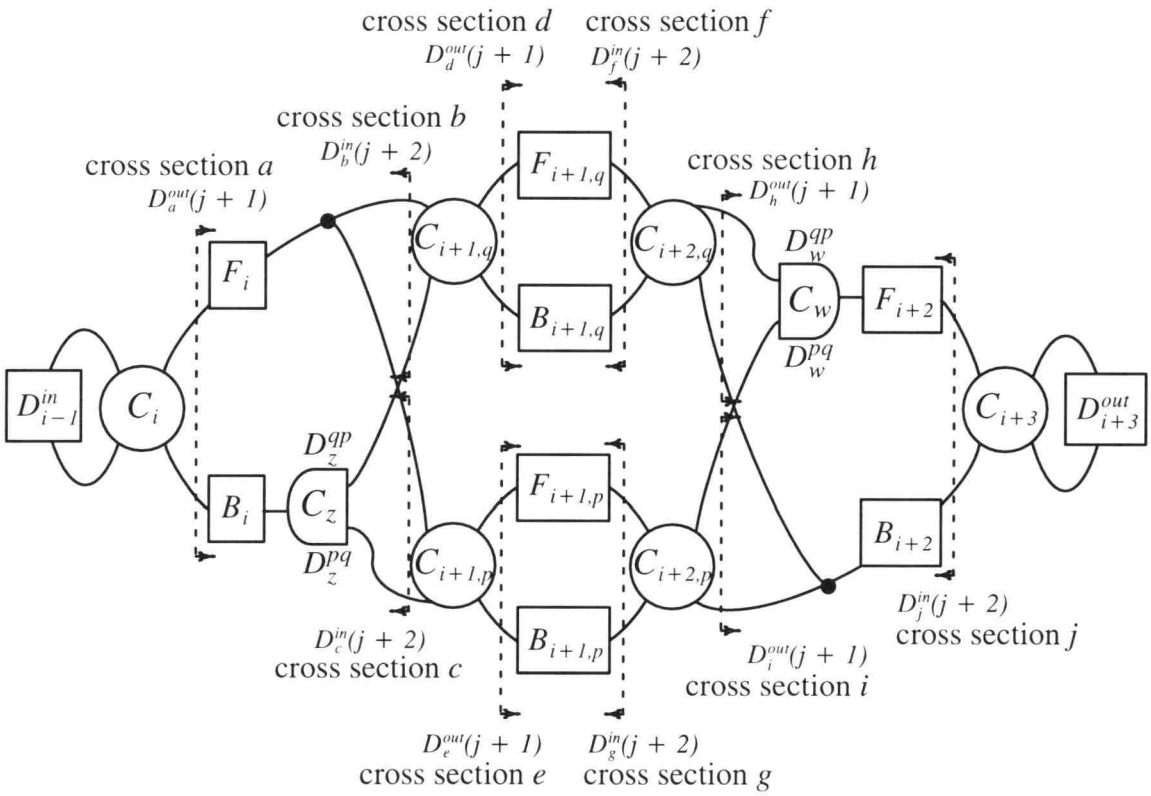


Figure 8.3: A two–dimensional pipeline system — closed loop.

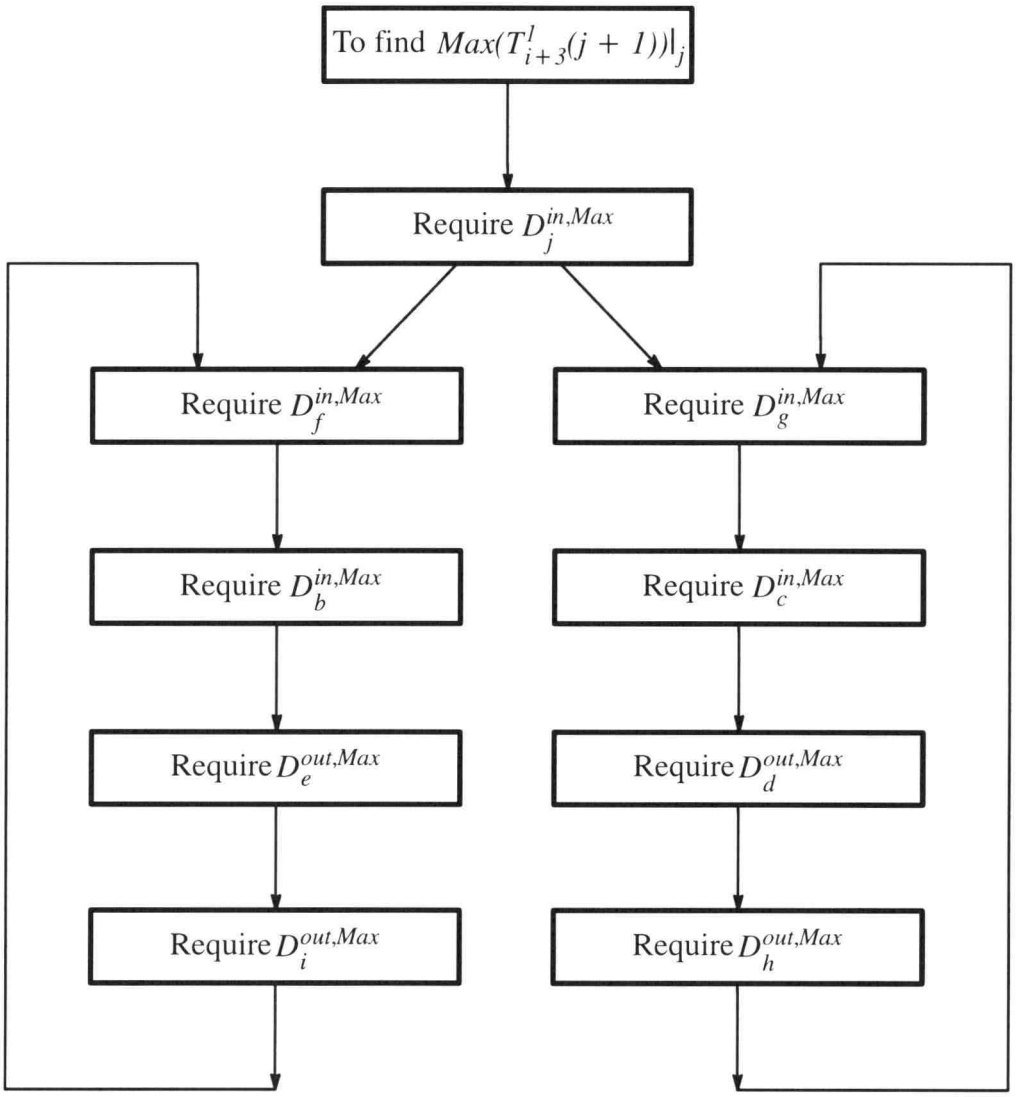


Figure 8.4: The dependency flow chart of Figure 8.3.

Using backward substitution for the expressions from (8.10) to (8.13), we conclude

$$
\begin{aligned}
& D_f^{in,Max} \\
& \le Max(F_{i+1,q}^{Max} + B_{i+1,q}^{Max}, \\
& \qquad F_i^{Max} + B_i^{Max} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}, \\
& \qquad F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}, \\
& \qquad 2F_{i+1,q}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} + B_{i+2}^{Max} + B_{i+1,p}^{Max} - B_{i+1,p}^{min} + F_i^{Max} - F_i^{min} \\
& \qquad\qquad + B_i^{Max} - B_i^{min} - F_{i+1,q}^{min}, \\
& \qquad F_{i+2}^{Max} + B_{i+2}^{Max} + B_{i+1,p}^{Max} - B_{i+1,p}^{min} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} \\
& \qquad\qquad + F_{i+1,q}^{Max} - F_{i+1,q}^{min}, \\
& \qquad D_{i+3}^{out,Max} + B_{i+2}^{Max} - B_{i+2}^{min} + B_{i+1,p}^{Max} - B_{i+1,p}^{min} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} \\
& \qquad\qquad + F_{i+1,q}^{Max} - F_{i+1,q}^{min}, \\
& \qquad D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}, \\
& \qquad D_f^{in,Max} + F_{i+2}^{Max} - F_{i+2}^{min} + B_{i+2}^{Max} - B_{i+2}^{min} + B_{i+1,p}^{Max} - B_{i+1,p}^{min} + F_i^{Max} \\
& \qquad\qquad - F_i^{min} + B_i^{Max} - B_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}) \qquad (8.14)
\end{aligned}
$$

From (8.14), by looking into the very last element, the value of $F_{i+2}^{Max} - F_{i+2}^{min} + B_{i+2}^{Max} - B_{i+2}^{min} + B_{i+1,p}^{Max} - B_{i+1,p}^{min} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min}$ is always positive. This suggests that $D_f^{in,Max}$ could be infinite since the statement of $D_f^{in,Max} \le D_f^{in,Max}$ + (positive number) always holds for arbitrary value of $D_f^{in,Max}$. However, $D_f^{in,Max}$ can not be infinite in this system. The reason why (8.14) shows the possibility of having infinite value of $D_f^{in,Max}$ is because logic delay terms which have preceding "–" sign are dropped in this expression. Instead of bringing logic delay terms back into (8.14) (since this will lead to a very complex expression), one more step is employed

before regular approach is applied. This extra step is to find a relationship of the "loop" and then break this loop down.

By looking into Figure 8.5, two different loops are observed. One is a data–token loop which is formed by two data token traveling paths, marked by black solid and gray solid lines. The other is a space–token loop which is constructed by two space token routing paths, highlighted by black dash and gray dash lines. Apparently, the time spending on these paths for these two data tokens should be the same. That is,

$$D^{24}_{i+1,q}(j+1) + F_{i+1,q}(j+1) + D^{24}_{i+2,q}(j+1) + D^{qp}_{w}(j+1)$$

$$= D^{24}_{i+1,p}(j+1) + F_{i+1,p}(j+1) + D^{24}_{i+2,p}(j+1) + D^{pq}_{w}(j+1) \tag{8.15}$$

$$\Rightarrow D^{24}_{i+1,q}(j+2) + F_{i+1,q}(j+2) + D^{24}_{i+2,q}(j+2) + D^{qp}_{w}(j+2)$$

$$= D^{24}_{i+1,p}(j+2) + F_{i+1,p}(j+2) + D^{24}_{i+2,p}(j+2) + D^{pq}_{w}(j+2) \tag{8.16}$$

Similarly, the traveling time for these two space tokens should be the same. That is,

$$D^{42}_{i+2,q}(j+2) + B_{i+1,q}(j+2) + D^{42}_{i+1,q}(j+3) + D^{qp}_{z}(j+3)$$

$$= D^{42}_{i+2,p}(j+2) + B_{i+1,p}(j+2) + D^{42}_{i+1,p}(j+3) + D^{pq}_{z}(j+3) \tag{8.17}$$

If the Equal loop–delay theorem is applied to the C–element C_w, we have

$$D^{42}_{i+2,q}(j+2) + D^{qp}_{w}(j+2) = D^{42}_{i+2,p}(j+2) + D^{pq}_{w}(j+2) \tag{8.18}$$

Similarly, if the Equal loop–delay theorem is applied to the C–element C_z, we have

$$D^{24}_{i+1,q}(j+2) + D^{qp}_{z}(j+2) = D^{24}_{i+1,p}(j+2) + D^{pq}_{z}(j+2) \tag{8.19}$$

Note that $D^{24}_{i+1,q}(1) = D^{24}_{i+1,p}(1) = D^{qp}_{z}(1) = D^{pq}_{z}(1) = 0$.

If (8.16) – (8.18), we have

$$D^{24}_{i+1,q}(j+2) + F_{i+1,q}(j+2) + D^{24}_{i+2,q}(j+2) - D^{42}_{i+2,q}(j+2)$$

$$= D^{24}_{i+1,p}(j+2) + F_{i+1,p}(j+2) + D^{24}_{i+2,p}(j+2) - D^{42}_{i+2,p}(j+2) \tag{8.20}$$

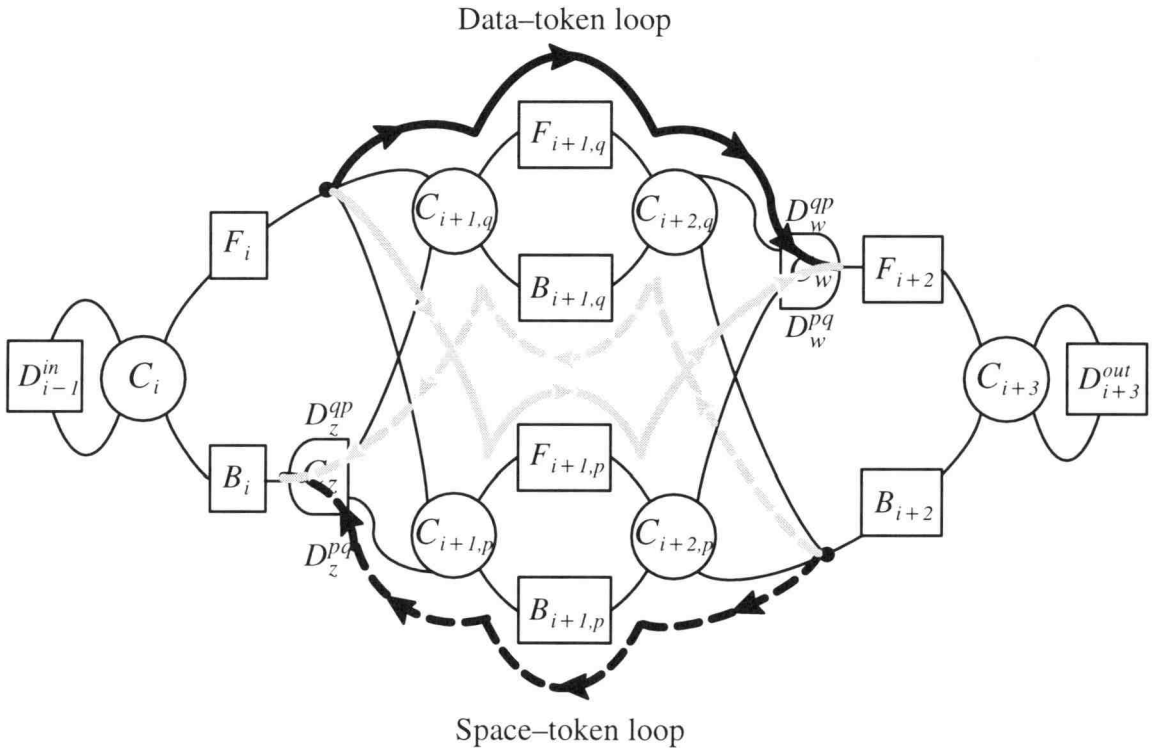


Figure 8.5: Two loops — one data–token loop and one space–token loop.

Also, from C–element's model, we know

$$D^{24}_{i+2,q}(j+2) * D^{42}_{i+2,q}(j+2) = 0 \tag{8.21}$$

$$D^{24}_{i+2,p}(j+2) * D^{42}_{i+2,p}(j+2) = 0 \tag{8.22}$$

From (8.20), (8.21) and (8.22), we can obtain $Max(T^{l}_{i+3}(j+1))|_j$ by discussing all of the possible cases as will be shown below.

By rearranging (8.20), we obtain

$$\begin{aligned} & D^{24}_{i+2,q}(j+2) \\ & = F_{i+1,p}(j+2) - F_{i+1,q}(j+2) + D^{24}_{i+1,p}(j+2) + D^{24}_{i+2,p}(j+2) + D^{42}_{i+2,q}(j+2) \\ & \qquad - D^{24}_{i+1,q}(j+2) - D^{42}_{i+2,p}(j+2) \end{aligned} \tag{8.23}$$

Substituting (8.23) into (8.21), we have

$D^{42}_{i+2,q}(j+2) = 0$ or

$$\begin{aligned} D^{42}_{i+2,q}(j+2) = {} & F_{i+1,q}(j+2) - F_{i+1,p}(j+2) + D^{24}_{i+1,q}(j+2) + D^{42}_{i+2,p}(j+2) \\ & - D^{24}_{i+1,p}(j+2) - D^{24}_{i+2,p}(j+2) \end{aligned} \tag{8.24}$$

Similarly, by rearranging (8.20), we obtain

$$\begin{aligned} & D^{24}_{i+2,p}(j+2) \\ & = F_{i+1,q}(j+2) - F_{i+1,p}(j+2) + D^{24}_{i+1,q}(j+2) + D^{42}_{i+2,p}(j+2) + D^{24}_{i+2,q}(j+2) \\ & \qquad - D^{24}_{i+1,p}(j+2) - D^{42}_{i+2,q}(j+2) \end{aligned} \tag{8.25}$$

Substituting (8.25) into (8.22), we have

$D^{42}_{i+2,p}(j+2) = 0$ or

$$\begin{aligned} D^{42}_{i+2,p}(j+2) = {} & F_{i+1,p}(j+2) - F_{i+1,q}(j+2) + D^{24}_{i+1,p}(j+2) + D^{42}_{i+2,q}(j+2) \\ & - D^{24}_{i+1,q}(j+2) - D^{24}_{i+2,q}(j+2) \end{aligned} \tag{8.26}$$

From (8.24) and (8.26), four cases are possible as described below.

Case 1>: $D^{42}_{i+2,q}(j+2) = 0$ and $D^{42}_{i+2,p}(j+2) = 0$

In this case, this implies $D_w^{qp}(j+2) = D_w^{pq}(j+2) = 0$. Hence, the equivalent circuit is drawn in Figure 8.6(a). The $Max(T_{i+3}^{1}(j+1))|_j$ has the form of

$$Max(T_{i+3}^{1}(j+1))|_j = Max(D_{i+3}^{out,Max}, F_{i+2}^{Max} + B_{i+2}^{Max}) \tag{8.27}$$

Case 2>: $D_{i+2,q}^{42}(j+2) = 0$ and

$$D_{i+2,p}^{42}(j+2) = F_{i+1,p}(j+2) - F_{i+1,q}(j+2) + D_{i+1,p}^{24}(j+2) - D_{i+1,q}^{24}(j+2) - D_{i+2,q}^{24}(j+2) \tag{8.28}$$

Since $D_{i+2,q}^{42}(j+2) = 0$, from (8.18) and the fact of $D_w^{qp}(j+2) * D_w^{pq}(j+2) = 0$, we conclude $D_w^{pq}(j+2) = 0$ and $D_w^{qp}(j+2) = D_{i+2,p}^{42}(j+2)$. The equivalent circuit in this case is drawn in Figure 8.6(b). As long as $D_z^{pq}(j+1)$ ($D_z^{pq}(1) = 0$) is derived, $Max(T_{i+3}^{1}(j+1))|_j$ can be obtained. It is the output loop delay of the FIFO with the backward delay of the first stage being the sum of $B_i(j+1) + D_z^{pq}(j+1)$. From (8.19), we know

$$D_z^{pq}(j+2) = Max(0, D_{i+1,q}^{24}(j+2) - D_{i+1,p}^{24}(j+2)) \tag{8.29}$$

From (8.28), we have

$$D_{i+1,q}^{24}(j+2) - D_{i+1,p}^{24}(j+2) = F_{i+1,p}(j+2) - F_{i+1,q}(j+2) - D_{i+2,q}^{24}(j+2) - D_{i+2,p}^{42}(j+2) \tag{8.30}$$

Combining (8.29) and (8.30), we obtain

$$D_z^{pq}(j+2) = Max(0, F_{i+1,p}(j+2) - F_{i+1,q}(j+2) - D_{i+2,q}^{24}(j+2) - D_{i+2,p}^{42}(j+2))$$

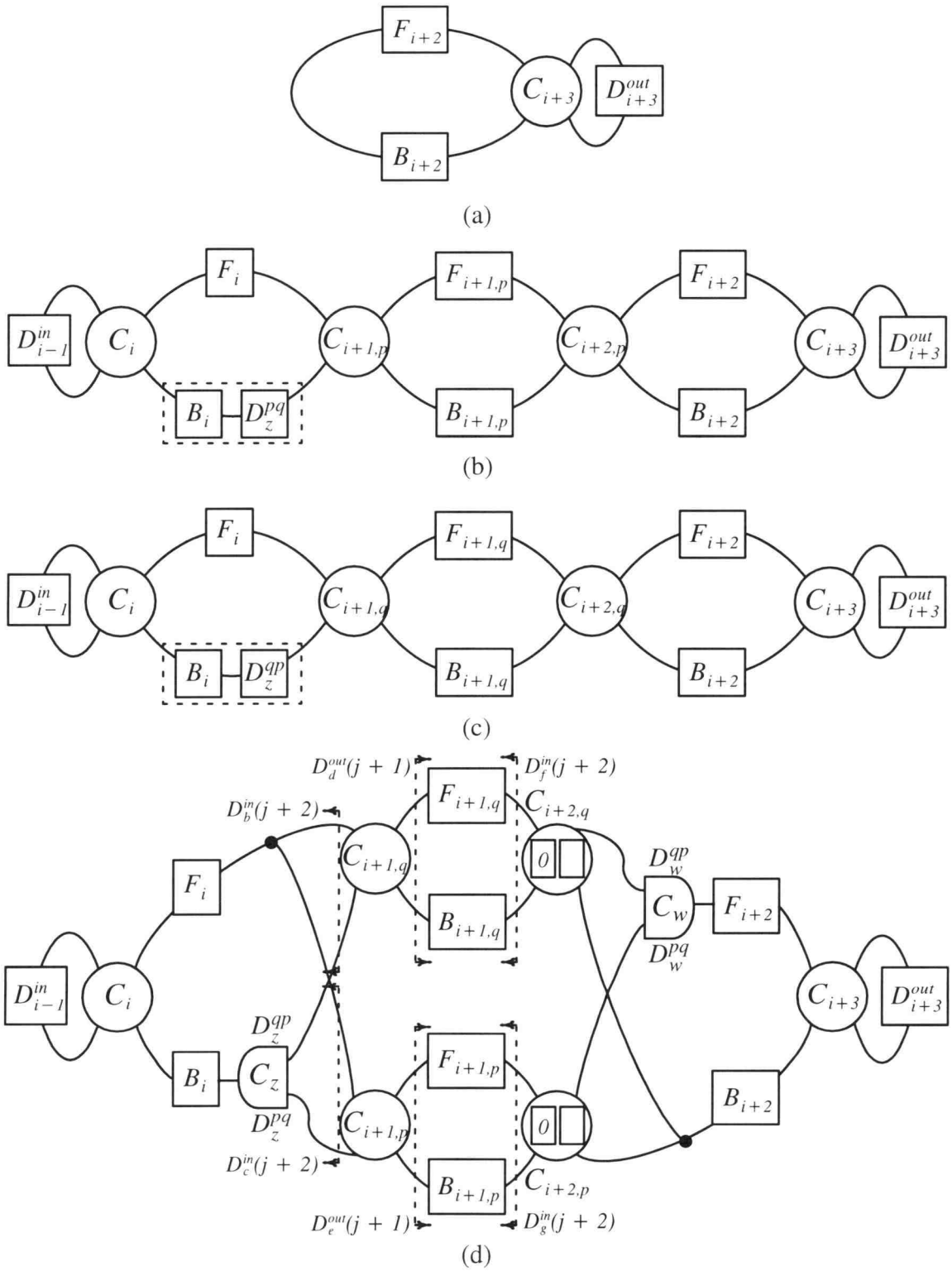


Figure 8.6: Equivalent circuits of Figure 8.3 corresponding to different cases.
(a) Equivalent circuit corresponding to case *1*.
(b) Equivalent circuit corresponding to case *2*.
(c) Equivalent circuit corresponding to case *3*.
(d) Equivalent circuit corresponding to case *4*.

$$\leq Max(0,$$

$$F_{i+1,p}(j+2) - F_{i+1,q}(j+2))$$

Therefore,

$$B_i(j+1) + D_z^{pq}(j+1) \leq Max(B_i^{Max},$$

$$B_i^{Max} + F_{i+1,p}^{Max} - F_{i+1,q}^{min}) \tag{8.31}$$

From Figure 8.6(b), we have

$$Max(T_{i+3}^{l}(j+1))|_j$$

$$\leq Max(D_{i+3}^{out,Max},$$

$$F_{i+2}^{Max} + B_{i+2}^{Max},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + 2F_{i+1,p}^{Max} - F_{i+1,p}^{min} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min}) \tag{8.32}$$

Case 3>: $D_{i+2,p}^{42}(j+2) = 0$ and

$$D_{i+2,q}^{42}(j+2) = F_{i+1,q}(j+2) - F_{i+1,p}(j+2) + D_{i+1,q}^{24}(j+2)$$

$$- D_{i+1,p}^{24}(j+2) - D_{i+2,p}^{24}(j+2)$$

In this case, due to the symmetry of this circuit, we can obtain $Max(T_{i+3}^{l}(j+1))|_j$ by changing p with q and q with p in (8.32). That is,

$$Max(T_{i+3}^{l}(j+1))|_j$$

$$\leq Max(D_{i+3}^{out,Max},$$

$$F_{i+2}^{Max} + B_{i+2}^{Max},$$

$$F_{i+1,q}^{Max} + B_{i+1,q}^{Max} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + F_{i+1,q}^{Max} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + 2F_{i+1,q}^{Max} - F_{i+1,q}^{min} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min}) \tag{8.33}$$

The equivalent circuit corresponding to this case is shown in Figure 8.6(c).

Case 4>: $D_{i+2,p}^{42}(j+2) = F_{i+1,p}(j+2) - F_{i+1,q}(j+2) + D_{i+1,p}^{24}(j+2) + D_{i+2,q}^{42}(j+2)$

$$- D_{i+1,q}^{24}(j+2) - D_{i+2,q}^{24}(j+2) \text{ and}$$

$$D_{i+2,q}^{42}(j+2) = F_{i+1,q}(j+2) - F_{i+1,p}(j+2) + D_{i+1,q}^{24}(j+2) + D_{i+2,p}^{42}(j+2)$$

$$- D_{i+1,p}^{24}(j+2) - D_{i+2,p}^{24}(j+2)$$

In this case, if the above two equations are added together, the resulting equation becomes $D_{i+2,q}^{24}(j+2) + D_{i+2,p}^{24}(j+2) = 0$. Since all logic delays, according to our model, are greater than or equal to zero, $D_{i+2,q}^{24}(j+2) = D_{i+2,p}^{24}(j+2) = 0$ in this case. The equivalent circuit corresponding to this case is depicted in Figure 8.6(d). Since $D_{i+2,q}^{24}(j+2) = D_{i+2,p}^{24}(j+2) = 0$ and $D_{i+2,q}^{24}(1) = D_{i+2,p}^{24}(1) = 0$, we have

$$D_d^{out}(j+1) = F_{i+1,q}(j+1) + B_{i+1,q}(j+1)$$

$$\Rightarrow D_d^{out,max} = F_{i+1,q}^{Max} + B_{i+1,q}^{Max} \tag{8.34}$$

$$D_e^{out}(j+1) = F_{i+1,p}(j+1) + B_{i+1,p}(j+1)$$

$$\Rightarrow D_e^{out,max} = F_{i+1,p}^{Max} + B_{i+1,p}^{Max} \tag{8.35}$$

By looking into the Fork in Figure 8.6(d) and from (8.34) and (8.35), $D_b^{in,max}$ and $D_c^{in,max}$ can be obtained. That is,

$$D_b^{in,max} \le Max(F_i^{Max} + B_i^{Max},$$

$$D_e^{out,max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min})$$

$$\le Max(F_i^{Max} + B_i^{Max},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \tag{8.36}$$

Similarly, we have

$$D_c^{in,max} \leq Max(F_i^{Max} + B_i^{Max},$$

$$F_{i+1,q}^{Max} + B_{i+1,q}^{Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min}) \tag{8.37}$$

Now, due to the equivalent FIFO structure, $D_f^{in,Max}$ can be easily obtained based on (8.36). Same procedure can be applied to obtain $D_g^{in,Max}$. Once $D_f^{in,Max}$ and $D_g^{in,Max}$ become known, $Max(T_{i+3}^{l}(j+1))|_j$ can be approximated through Join structure. That is,

$$Max(T_{i+3}^{l}(j+1))|_j$$

$$\leq Max(D_{i+3}^{out,Max},$$

$$F_{i+2}^{Max} + B_{i+2}^{Max},$$

$$F_{i+1,q}^{Max} + B_{i+1,q}^{Max} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + F_{i+1,q}^{Max} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min} + F_{i+1,q}^{Max} - F_{i+1,q}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_i^{Max} + B_i^{Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$F_{i+1,p}^{Max} + B_{i+1,p}^{Max} + F_i^{Max} - F_i^{min} + B_i^{Max} - B_i^{min} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min},$$

$$D_{i-1}^{in,Max} + F_i^{Max} - F_i^{min} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} + F_{i+2}^{Max} - F_{i+2}^{min}) \tag{8.38}$$

As the result of discussing four different cases for the circuit of Figure 8.3, its maximum output loop delay could be chosen as the maximum of (8.27), (8.32), (8.33) and (8.38). That is,

$$
\begin{aligned}
& Max(T^{l}_{i+3}(j+1))|_{j} \\
& \leq Max(D^{out,Max}_{i+3}, \\
& \quad F^{Max}_{i+2} + B^{Max}_{i+2}, \\
& \quad F^{Max}_{i+1,q} + B^{Max}_{i+1,q} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i} + B^{Max}_{i} + F^{Max}_{i+1,q} - F^{min}_{i+1,q} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i+1,p} + B^{Max}_{i+1,p} + F^{Max}_{i} - F^{min}_{i} + B^{Max}_{i} - B^{min}_{i} + F^{Max}_{i+1,q} - F^{min}_{i+1,q} \\
& \qquad + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i} + B^{Max}_{i} + 2F^{Max}_{i+1,q} - F^{min}_{i+1,q} - F^{min}_{i+1,p} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad D^{in,Max}_{i-1} + F^{Max}_{i} - F^{min}_{i} + F^{Max}_{i+1,q} - F^{min}_{i+1,q} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i+1,p} + B^{Max}_{i+1,p} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i} + B^{Max}_{i} + F^{Max}_{i+1,p} - F^{min}_{i+1,p} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i+1,p} + B^{Max}_{i+1,p} + F^{Max}_{i} - F^{min}_{i} + B^{Max}_{i} - B^{min}_{i} + F^{Max}_{i+1,p} - F^{min}_{i+1,p} \\
& \qquad + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad F^{Max}_{i} + B^{Max}_{i} + 2F^{Max}_{i+1,p} - F^{min}_{i+1,p} - F^{min}_{i+1,q} + F^{Max}_{i+2} - F^{min}_{i+2}, \\
& \quad D^{in,Max}_{i-1} + F^{Max}_{i} - F^{min}_{i} + F^{Max}_{i+1,p} - F^{min}_{i+1,p} + F^{Max}_{i+2} - F^{min}_{i+2})
\end{aligned}
\tag{8.39}
$$

Several simulations are done to compare our approximations based on (8.39) with the exact values obtained through Petri net model and CTSE simulation. The result is summarized in Table 8.2.

Table 8.2: Comparison of our approximations with the exact bounds using Figure 8.3 (closed–loop system) as an example.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
D^{in}	[6 10]	[2 15]	[2 5]	[12 15]	[12 25]	[12 15]	[2 5]	[10 12]	[10 12]	[10 32]
F_i	[4 9]	[4 11]	[14 19]	[4 19]	[4 9]	[4 9]	[4 9]	[9 18]	[2 8]	[12 18]
B_i	[3 5]	[13 28]	[3 8]	[3 18]	[3 18]	[3 8]	[3 8]	[4 9]	[4 9]	[4 9]
$F_{i+1,q}$	[25 28]	[9 10]	[9 10]	[19 30]	[5 10]	[5 10]	[5 20]	[5 12]	[5 9]	[5 11]
$B_{i+1,q}$	[13 15]	[16 18]	[6 8]	[6 8]	[6 8]	[6 8]	[6 8]	[4 7]	[4 7]	[4 7]
$F_{i+1,p}$	[20 30]	[10 20]	[1 8]	[10 28]	[5 18]	[5 8]	[5 8]	[9 18]	[9 15]	[4 10]
$B_{i+1,p}$	[19 25]	[13 18]	[3 10]	[3 5]	[3 5]	[3 5]	[3 5]	[9 12]	[9 12]	[11 12]
F_{i+2}	[24 29]	[11 20]	[5 10]	[5 15]	[5 15]	[5 10]	[5 15]	[7 10]	[7 10]	[7 9]
B_{i+2}	[23 28]	[3 18]	[3 5]	[3 5]	[3 5]	[3 5]	[3 15]	[9 25]	[5 10]	[5 9]
D^{out}	[19 25]	[9 25]	[39 55]	[29 45]	[5 10]	[15 60]	[15 40]	[15 30]	[5 10]	[15 20]
Max	60	59	55	58	53	60	42	39	30	46
UP_1	70	70	55	96	61	60	51	54	45	46

The results shown in Table 8.2 show that the approximations for a closed–loop system can be over one and half times larger than the exact value. These results are not as close to the exact value as the results derived for the open–loop system (see Table 8.1) mentioned in previous section. However, if we treat the whole closed–loop system as a basic pipeline module and derive the results by applying the Equal loop–delay theorem to all C–elements in this system (as done in Chapter 7), we may be able to obtain a result even closer to the exact value. This is because when the Equal loop–delay theorem is applied, the relationship or dependency among logic delays, forward delays and backward delays can be revealed through difference equations. This dependency will bring a result closer to the exact value. For example, let

$$A(j+1) = F(j+3) - F(j+2) \tag{8.40}$$

$$B(j+1) = F(j+2) - F(j+1) \tag{8.41}$$

where $F^{min} \leq F(j+1) \leq F^{Max}$

Find the upper bound of $A(j+1) + B(j+1)$.

Approach 1> If we consider the dependency of $A(j+1)$ and $B(j+1)$, we have

$$A(j+1) + B(j+1) = F(j+3) - F(j+2) + F(j+2) - F(j+1) = F(j+3) - F(j+1)$$

$$\Rightarrow Max(A(j+1) + B(j+1))|_j = F^{Max} - F^{min} \tag{8.42}$$

Approach 2> If we do not take dependency into account, we have

$$A(j+1) = F(j+3) - F(j+2)$$

$$\Rightarrow Max(A(j+1))|_j = F^{Max} - F^{min} \tag{8.43}$$

$$B(j+1) = F(j+2) - F(j+1)$$

$$\Rightarrow Max(B(j+1))|_j = F^{Max} - F^{min} \tag{8.44}$$

From (8.42), (8.43) and (8.44), we have

$$\begin{aligned} Max(A(j+1) + B(j+1))|_j &\leq Max(A(j+1))|_j + Max(B(j+1))|_j \\ &\leq F^{Max} - F^{min} + F^{Max} - F^{min} \\ &\leq 2(F^{Max} - F^{min}) \end{aligned} \tag{8.45}$$

From (8.42) and (8.45), we know there is a $F^{Max} - F^{min}$ difference between these two approaches.

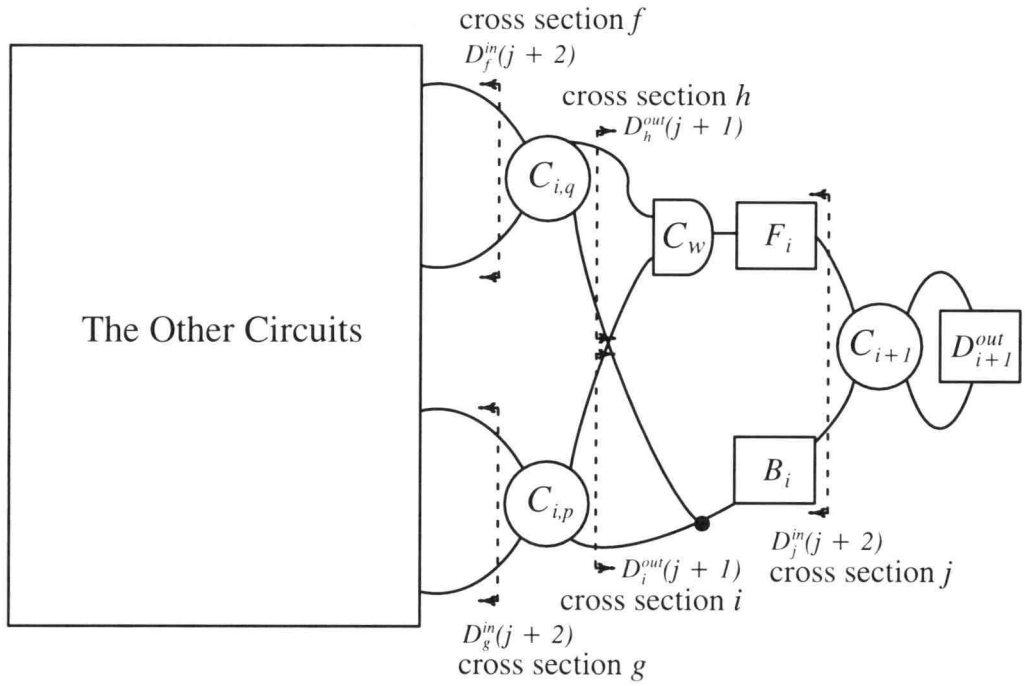
The other kind of dependency also affects the accuracy of the approximation. When each module was discussed in Chapter 7, all of the physical delays, including environmental input and output delays, forward and backward delays were assumed to be independent. For example, in Join module (Figure 7.3) $D_q^{in}(j+2)$ and $D_p^{in}(j+2)$ are independent of $F_t(j+1)$ and $B_t(j+1)$. This assumption is valid when an open–loop system is considered. It is, however, not true for a closed–loop system. For example, when considering the Join module in Figure 8.3, the environmental input delays will be $D_f^{in}(j+2)$ and $D_g^{in}(j+2)$. Also, from Figure 8.4, it is noticed that $D_f^{in,Max}$ is a function of $D_i^{out,Max}$, and $D_g^{in,Max}$ is a function of $D_h^{out,Max}$. These imply that the environmental

delay bounds depend on stage–delays F_{i+2}^{Max}, F_{i+2}^{min}, B_{i+2}^{Max} or B_{i+2}^{min}, violating our assumption. The concept of this type of dependency can be further illustrated using a more general Join module, as shown in Figure 8.7. Based on expression (7.33), we have

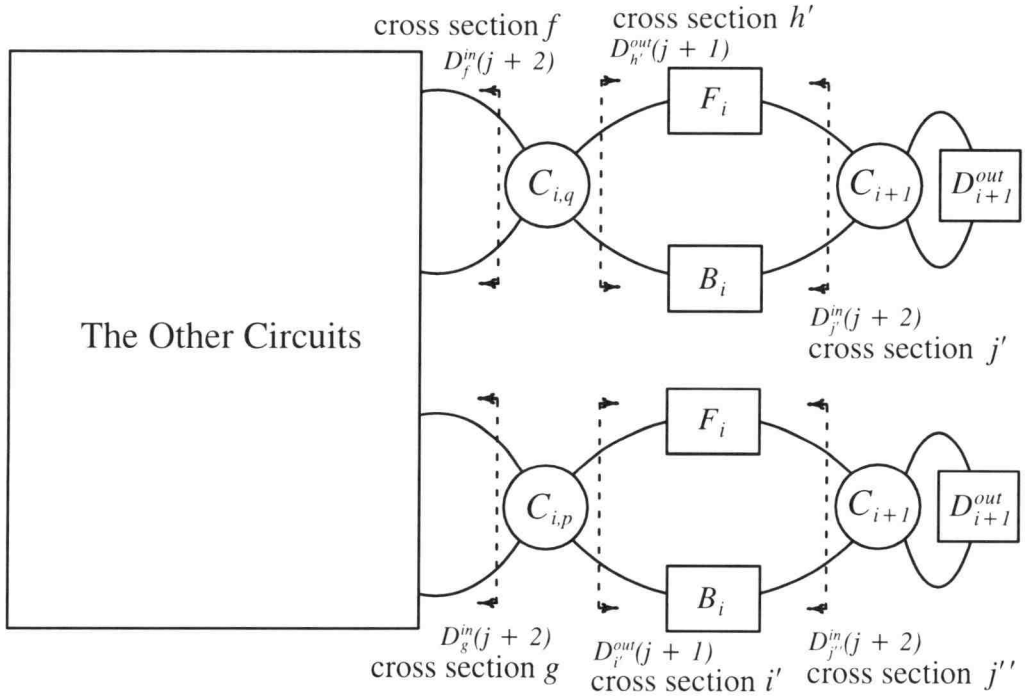
$$\begin{aligned} Max(T_{i+1}^{l}(j+1))|_j \le Max(&D_{i+1}^{out,Max}, \\ &F_i^{Max} + B_i^{Max}, \\ &D_f^{in,Max} + F_i^{Max} - F_i^{min}, \\ &D_g^{in,Max} + F_i^{Max} - F_i^{min}) \end{aligned} \tag{8.46}$$

$$\begin{aligned} \Rightarrow Max(T_{i+1}^{l}(j+1))|_j \le Max(&Max(D_{i+1}^{out,Max}, \\ &F_i^{Max} + B_i^{Max}, \\ &D_f^{in,Max} + F_i^{Max} - F_i^{min}), \\ &Max(D_{i+1}^{out,Max}, \\ &F_i^{Max} + B_i^{Max}, \\ &D_g^{in,Max} + F_i^{Max} - F_i^{min})) \end{aligned} \tag{8.47}$$

Note that expression (8.46) is the approximation of Figure 8.7(a). Also, note that each of the components in (8.47) is equivalent to the maximum output loop delay representation of a FIFO. Therefore, (8.47) is equivalent to the maximum value of two maximum output loop delays corresponding to two separate FIFOs, as illustrated in Figure 8.7(b). As a result of the approximation equivalency of (8.46) and (8.47), the circuits in Figure 8.7(a) and Figure 8.7(b) are also equivalent as far as the maximum output loop delay (approximation) is concerned. However, caution should be taken when splitting a Join into two FIFOs. First, the system should be open loop. That is, the value of $D_f^{in,Max}$ is independent of $D_i^{out,Max}$ and the value of $D_g^{in,Max}$ is independent of $D_h^{out,Max}$. Second, this conclusion is applicable to obtain $Max(T_{i+1}^{l}(j+1))|_j$ only and nothing else. Third, $D_h^{out,Max}$ is different from $D_{h'}^{out,Max}$ and $D_i^{out,Max}$ is different from $D_{i'}^{out,Max}$. For



(a)



(b)

Figure 8.7: A general system with Join module and its approximation equivalent.
(a) A general system with Join module.
(b) Approximation equivalent if the system is open–loop.

example, in Figure 8.1, to find $Max(T^{l}_{i+2,p}(j+1))|_j$, the split of Join module into two FIFOs is valid since the system is open loop. The result will be the maximum value of these two individual maximum output loop delay. However, to find the value of $Max(T^{l}_{i+2,t}(j+1))|_j$ and $Max(T^{l}_{i+2,u}(j+1))|_j$, splitting the Join module is disallowed. This is because $D_a^{in,Max}$ is required in order to find $Max(T^{l}_{i+2,t}(j+1))|_j$ and $Max(T^{l}_{i+2,u}(j+1))|_j$ and $D_d^{out,Max}$ is related to $D_a^{in,Max}$ when Toggle/XOR module is considered. Based on the caution noted above, $D_d^{out,Max}$ will not be the same before and after the Join is split. Therefore the approximation equivalency does not hold when $Max(T^{l}_{i+2,t}(j+1))|_j$ and $Max(T^{l}_{i+2,u}(j+1))|_j$ are calculated.

8.3 A Design Example

Our approach provides both a method for analyzing the performance of asynchronous circuits and guidelines (or approaches) for designing such circuits. This is achievable because we use a symbolic, rather than a numerical, approach. The characteristics of our method allow designers to analyze circuit performance while, at the same time, providing design guidelines. A general FIFO example has been studied based on our approach in Chapter 5. The design guidelines provided when the FIFO example was discussed can also be applied to a more general pipeline circuit. In this section, we will use the open–loop circuit in Figure 8.1 as an example to demonstrate stage splitting, one of the design approaches.

Given a circuit architecture as shown in Figure 8.1, the delays are assumed to be bounded and described as follows.

$$10 \le D_q^{in}(j+2) \le 15,$$

$$10 \le D_p^{in}(j+2) \le 12,$$

$$4 \le F_{i,q}(j+1) \le 9,$$

$3 \le B_{i,q}(j+1) \le 8,$

$9 \le F_{i,p}(j+1) \le 30,$

$6 \le B_{i,p}(j+1) \le 8,$

$10 \le F_{i+1,r}(j+1) \le 18,$

$3 \le B_{i+1,r}(j+1) \le 5,$

$5 \le F_{i+1,p}(j+1) \le 15,$

$3 \le B_{i+1,p}(j+1) \le 5,$

$9 \le D_t^{out}(j+1) \le 15,$

$6 \le D_u^{out}(j+1) \le 10,$ and

$2 \le D_p^{out}(j+1) \le 9.$

With the delay bounds shown above, the result of output loop delays for $T_{i+2,t}^{l}(j+1)$, $T_{i+2,u}^{l}(j+1)$ and $T_{i+2,p}^{l}(j+1)$ can be obtained through (8.1) to (8.8). That is,

1. $$\begin{aligned} D_c^{out,Max} &\le Max(18+5, \\ &\qquad 10+5-3, \\ &\qquad 15+5-3) \\ &\le Max(23,\ 12,\ 17) \le 23 \end{aligned} \tag{8.48}$$

$$\begin{aligned} D_f^{in,Max} &= Max(30+8, \\ &\qquad 12+30-9) \\ &= Max(38,\ 33) = 38 \end{aligned} \tag{8.49}$$

2. $$\begin{aligned} D_b^{in,Max} &\le Max(2*9+2*8, \\ &\qquad 15+2*9-4+8, \\ &\qquad 23+9-4+8-3, \\ &\qquad 2*15+9-4) \\ &\le Max(34,\ 37,\ 33,\ 35) \le 37 \end{aligned} \tag{8.50}$$

3. $D_d^{out,Max} \leq Max(15 + 5,$

$12 + 30 - 10 - 2*4 - 3 + 15 + 5,$

$38 + 15 - 5 + 5 - 3,$

$9 + 5 - 3)$

$\leq Max(20, 41, 50, 11) \leq 50$ (8.51)

$\Rightarrow Max(T_{i+2,p}^{l}(j + 1))|_j \leq Max(9,$

$15 + 5,$

$37 + 15 - 5,$

$38 + 15 - 5)$

$\leq Max(9, 20, 47, 48) \leq 48$ (8.52)

4. $D_a^{in,Max} \leq Max(2*9 + 2*8,$

$15 + 2*9 - 4 + 8,$

$50 + 9 - 4 + 8 - 3,$

$2*15 + 9 - 4)$

$\leq Max(34, 37, 60, 35) \leq 60$ (8.53)

$\Rightarrow Max(T_{i+2,t}^{l}(j + 1))|_j \leq Max(15,$

$18 + 5,$

$10 + 18 - 10 + 5 - 3,$

$60 + 18 - 10)$

$\leq Max(15, 23, 20, 68) \leq 68$ (8.54)

$\Rightarrow Max(T_{i+2,u}^{l}(j + 1))|_j \leq Max(10,$

$18 + 5,$

$15 + 18 - 10 + 5 - 3,$

$60 + 18 - 10)$

$\leq Max(10, 23, 25, 68) \leq 68$ (8.55)

We conclude

$$Max(T^{l}_{i+2,t}(j+1))|_j = UP_l(T^{l}_{i+2,t}(j+1))|_j = 68.$$

$$Max(T^{l}_{i+2,u}(j+1))|_j = UP_l(T^{l}_{i+2,u}(j+1))|_j = 68.$$

$$Max(T^{l}_{i+2,p}(j+1))|_j = UP_l(T^{l}_{i+2,p}(j+1))|_j = 48.$$

For this specific design example, if the specification of $Max(T^{l}_{i+2,p}(j+1))|_j$ is changed to a smaller number, two options, shown in Figure 8.8(a), are available, according to the maximum numbers in (8.49) and (8.52).

1. Reduce $D_f^{in,Max}$: $D_f^{in,Max}$ can be reduced by splitting the FIFO stage. Based on our assumption and discussion of FIFO (Chapter 5), $D_f^{in,Max}$ is limited to a minimum value of *33* no matter how many stages it is split into.

2. Split Join: The intention of splitting the Join stage is to reduce the value of $F^{Max}_{i+1,p} - F^{min}_{i+1,p}$. Assume that $F_{i+1,p}$ and $B_{i+1,p}$ are split into two stages and the resulting pipeline has the following delay relationship. That is,

$$F^{Max}_{i+1,p} = F^{Max}_{i+1,p1} + F^{Max}_{i+1,p2},$$

$$F^{min}_{i+1,p} = F^{min}_{i+1,p1} + F^{min}_{i+1,p2},$$

$$B^{Max}_{i+1,p} = B^{Max}_{i+1,p1} + B^{Max}_{i+1,p2},$$

$$B^{min}_{i+1,p} = B^{min}_{i+1,p1} + B^{min}_{i+1,p2}.$$

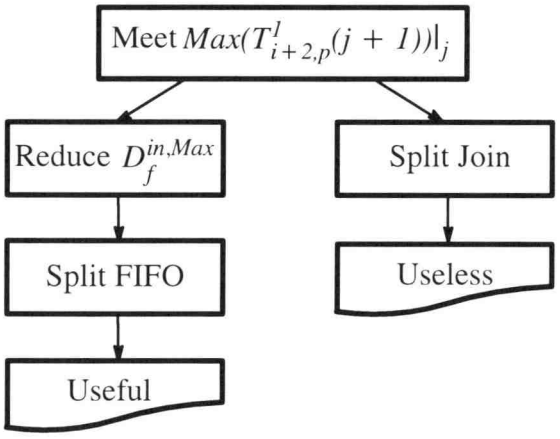
The new $Max(T^{l}_{i+2,p}(j+1))|_j$ becomes

$$Max(T^{l}_{i+2,p}(j+1))|_j$$

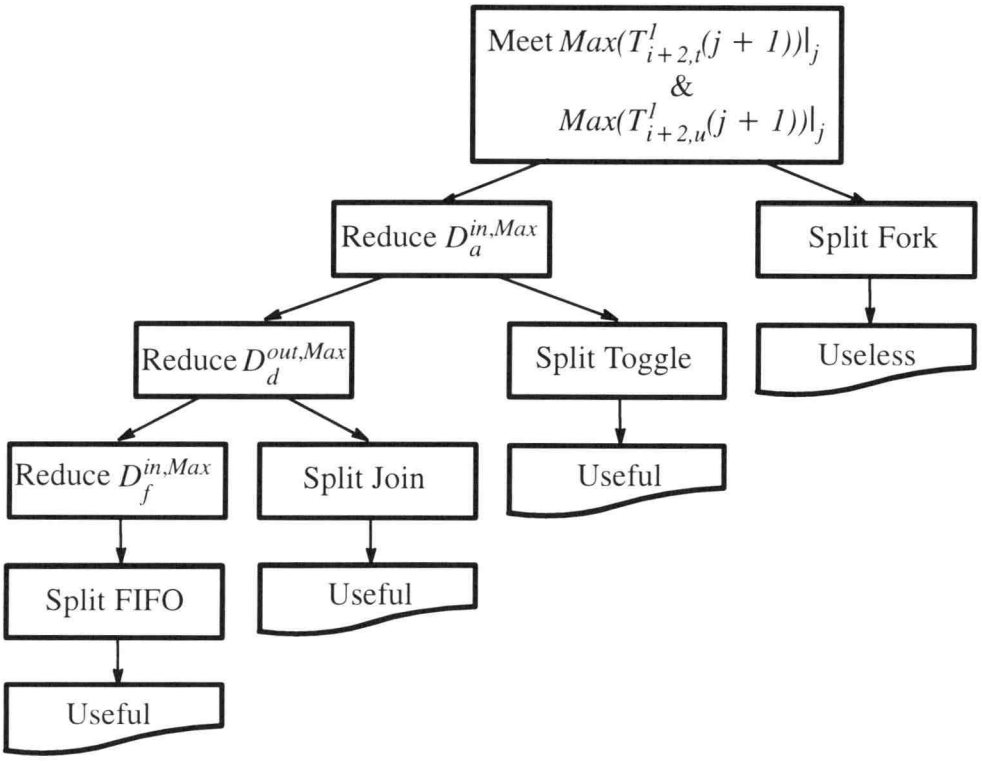
$$\leq Max(D_p^{out,Max},$$

$$F^{Max}_{i+1,p2} + B^{Max}_{i+1,p2}$$

$$F^{Max}_{i+1,p1} + B^{Max}_{i+1,p1} + F^{Max}_{i+1,p2} - F^{min}_{i+1,p2},$$



(a)



(b)

Figure 8.8: Design options for helping reach specifications.
(a) Flow chart for the improving of $Max(T^{l}_{i+2,p}(j+1))|_j$.
(b) Flow chart for the improving of $Max(T^{l}_{i+2,t}(j+1))|_j$ and $Max(T^{l}_{i+2,u}(j+1))|_j$.

$$D_b^{in,Max} + F_{i+1,p1}^{Max} - F_{i+1,p1}^{min} + F_{i+1,p2}^{Max} - F_{i+1,p2}^{min},$$

$$D_f^{in,Max} + F_{i+1,p1}^{Max} - F_{i+1,p1}^{min} + F_{i+1,p2}^{Max} - F_{i+1,p2}^{min}) \quad (8.56)$$

The maximum element in (8.56) is

$$D_f^{in,Max} + F_{i+1,p1}^{Max} - F_{i+1,p1}^{min} + F_{i+1,p2}^{Max} - F_{i+1,p2}^{min}$$

$$= D_f^{in,Max} + F_{i+1,p}^{Max} - F_{i+1,p}^{min} \quad (8.57)$$

As shown in (8.57), the maximum element is the same before and after splitting the Join stage. Therefore, splitting the Join stage will not improve our approximation to $Max(T_{i+2,p}^{l}(j+1))|_j$ in this particular example. In conclusion, based on our approach, the minimum value of $Max(T_{i+2,p}^{l}(j+1))|_j$ is $(33 + 15 - 5) = 43$ no matter which or how many stages are split.

If the specification of $Max(T_{i+2,t}^{l}(j+1))|_j$ and $Max(T_{i+2,u}^{l}(j+1))|_j$ is changed to a smaller value, several design options, demonstrated in Figure 8.8(b), are available, according to the maximum value in each expression from (8.48) to (8.55).

1. Split Fork stage: The intention of splitting the Fork stage is to make the value of $F_{i+1,r}^{Max} - F_{i+1,r}^{min}$ smaller in order to have a smaller sum in (8.54) and (8.55). Again, using an approach similar to (8.56) and (8.57), splitting the Fork stage will not help in reducing the value of $Max(T_{i+2,t}^{l}(j+1))|_j$ and $Max(T_{i+2,u}^{l}(j+1))|_j$.

2. Reduce $D_a^{in,Max}$ by splitting the Toggle stage: This will help since $D_d^{out,Max}$ is fixed and the amount of the rest of the element $F_{i,q}^{Max} - F_{i,q}^{min} + B_{i,q}^{Max} - B_{i,q}^{min}$ will be reduced by splitting the Toggle stage.

3. Reduce $D_d^{out,Max}$ by splitting the Join stage: This will improve the overall sum of the maximum element in (8.51) since $D_f^{in,Max}$ is fixed and the amount of the rest of the element $F_{i+1,p}^{Max} - F_{i+1,p}^{min} + B_{i+1,p}^{Max} - B_{i+1,p}^{min}$ will be reduced by splitting the Join stage.

4. Reduce $D_f^{in,Max}$ by splitting the FIFO stage: This stage–splitting helps, as described before, but is limited to the minimum value of *33*.

For example, if the specification is changed to

$$Max(T^l_{i+2,t}(j+1))|_j \leq 64,$$

$$Max(T^l_{i+2,u}(j+1))|_j \leq 64,$$

$$Max(T^l_{i+2,p}(j+1))|_j \leq 47,$$

splitting the FIFO into two stages will help meet the new requirement. If we let the two new stages have

$$F^{min}_{i,p1} = 7,\ F^{Max}_{i,p1} = 20,\ F^{min}_{i,p2} = 2,\ F^{Max}_{i,p2} = 10, \text{ and}$$

$$B^{min}_{i,p1} = 5,\ B^{Max}_{i,p1} = 6,\ B^{min}_{i,p2} = 1,\ B^{Max}_{i,p2} = 2,$$

the new bounds become

$$Max(T^l_{i+2,t}(j+1))|_j = UP_l(T^l_{i+2,t}(j+1))|_j = 64.$$

$$Max(T^l_{i+2,u}(j+1))|_j = UP_l(T^l_{i+2,u}(j+1))|_j = 64.$$

$$Max(T^l_{i+2,p}(j+1))|_j = UP_l(T^l_{i+2,p}(j+1))|_j = 47.$$

Because our method is only an approximation, it should be noted that all of the design options stated above do not always reflect the real situation accordingly. In some cases, several terms dropped in our method may be taken into play in obtaining the maximum output loop delay. However, our approach is always conservative at all times. Also, if the Toggle is split and the maximum element is $2D_q^{in,Max} + F_{i,q}^{Max} - F_{i,q}^{min}$, extra effort should be made to obtain a more accurate result. The reason is because the coefficient *2* of $D_q^{in,Max}$ makes a worse approximation by ignoring the dependency between, for example, $D_q^{in}(j+2)$ and $D_q^{in}(j+3)$. Similar statements have been discussed in previous section. It is worth mentioning that, if the element with maximum value in any one of the expressions from (8.48) to (8.55) changes, the flow chart of design options (Figure 8.8) needs to be changed accordingly. For example, if the maximum element

equal to *60* being the third element in (8.53) is somehow changed to the second element due to different stage bounds, the design options must also change. In other words, the flow chart of design options is not only architecture dependent but also stage–delay dependent.

8.4 Summary

In this chapter, the performance of a larger system consisting of FIFO, Fork, Join, Toggle/XOR is discussed. The simulation result shows that our approach has a closer approximation for an open–loop system than for a closed–loop system. Our method provides not only the performance analysis, but also the design guidelines. The design guidelines (e.g., stage–splitting) do not always reflect the real circuit behavior, since our approach is only an approximation. However, they do give designers some clues for improving overall circuit performance in a systematic way, rather than on trial and error basis. Meanwhile, they also provide some information to save design cycle time (e.g., the stage–splitting for specific stages does not help in overall performance). It is best to use CTSE simulations together with this approach to verify the intermediate and final result. Finally, it should be noted that the design options are functions of both circuit architecture and stage–delays. That is, same circuit architecture may result in a different flow chart of design options if the stage–delays are different.

9. CONCLUSIONS AND FUTURE WORKS

The Micropipeline is one of many asynchronous design methodologies. It is also the design method we chose to study for its performance. One of the features that micropipelines have is modularity and composibility. We make use of this feature in our performance analysis and design. Conclusions are made in this chapter based on the result and discussion in previous chapters. A comparative summary of our approach and the CTSE is also included in the conclusions. Several future works based on and extending our approach are suggested at the end of this chapter.

9.1 Conclusions

There are two types of performance measurements in asynchronous circuits. One is the average throughput and the other is throughput bounds. Since stage–delay is totally random, it is not possible to find an average throughput from the deterministic approach adopted in this thesis. Instead, a probability approach is suggested. Even so, our approach reveals that not only total stage–delay (delay–sum), but also stage–delay pattern (asynchronous parameters — forward delay slope and logic delay) affects the average throughput. Based upon a finite number of input data, this leads to the conclusion that asynchronous pipelines may have better, worse or equivalent performance (average) than synchronous pipelines.

Most of this thesis is devoted to finding the throughput bounds for FIFO, Fork, Join, Toggle/XOR, Arbiter/Call, Select/XOR and a system composed of some of these modules. Our approach has two steps. First, several basic modules are chosen. They include FIFO, Fork, Join, Toggle/XOR, Arbiter/Call and Select/XOR. The output loop delay, equivalent input delay and equivalent output delay for each basic module are derived based on the Equal loop–delay theorem. The result is a set of difference equations. The performance approximation can be obtained with simple mathematical opera-

tion on the difference equations, given the bounds of stage–delays. That is, the bounds (to save space, lower bound is not derived for most of the modules) of output loop delay, equivalent input delay and equivalent output delay can be represented as the bounds of stage–delays. Second, for a larger system consisting of those basic modules, its performances bounds can be derived directly from the bounds of output loop delay, equivalent input delay and equivalent output delay of those basic modules, which have been obtained at first step. This approach allows a fast and easy calculation of performance bounds since re–deriving the difference equations for the whole system is avoided.

Our approach to the output loop delay can be from the left–hand or right–hand side (as when FIFO is discussed in Chapter 5). However, most of our efforts are devoted to approximating the maximum output loop delay from the right hand side, i.e., our approximation is greater than or equal to the exact maximum bound. The simulation result shows that our approach has a closer approximation for an open–loop system than for a closed–loop system. Since our method is a symbolic approach instead of a numerical approach, it allows designers to analyze circuit performance while providing design guidelines/approaches at the same time. These design guidelines/approaches do not always reflect the real circuit behavior, since our approach is only an approximation. However, they do give designers some clues for improving the overall circuit performance in a systematic way, rather than on trial and error basis. Meanwhile, they also provide some information which may save design cycle time. This thesis in not intended to replace the CTSE tool. On the contrary, it is best to use CTSE together with our method to verify the intermediate and final result. Also, it should be noted that design options are functions of both circuit architecture and stage–delays. That is, the same circuit architecture may result in different design options if the stage–delays are different.

In the design of an asynchronous circuit, there are many solutions (stage–delay bounds) that can meet the design specifications. Which solution is chosen depends on the current and available technology for implementing the circuit being designed. The

other factor that will affect the choice of stage–delays is the average throughput. Choosing different stage–delay bounds or changing stage–delay bounds to meet the specification of throughput bounds may reduce the average throughput. Therefore, the best approaches or choices to adopt are those that will make both average throughput and throughput bounds meet the requirements.

The following is a comparison summary of our approach and CTSE from many perspectives.

1. Approach

   CTSE: numerical

   Ours: symbolic

2. Algorithm

   CTSE: needs advanced mathematics and algorithm

   Ours: uses difference equations and simple operations

3. Application

   CTSE: general asynchronous circuits modeled with a Petri net

   Ours: micropipelines

4. Accuracy

   CTSE: exact bounds for circuits that are data–independent and without mutually exclusive mechanism

   Ours: an approximation, except for the maximum output loop delay of a initially reset FIFO

5. Modularity

   CTSE: the whole circuit should be simulated again whenever there is a change, even a small change on this circuit

   Ours: only the parts that are affected by this change need to be re–calculated

6. Design

CTSE: provides no direct information to help with design

Ours: provides design procedure and guidelines (approaches)

9.2 Future Works

Several extensions of this research and the results presented in this thesis are possible.

1.> Stage–delay dependency: In the problem set–up, we assume that all stage–delays are independent. That is, $F_p(j + 1)$ and $F_q(j + 1)$, $B_p(j + 1)$ and $B_q(j + 1)$, $F_p(j + 1)$ and $B_p(j + 1)$, and $F_p(j + 1)$ and $B_q(j + 1)$ are independent where $p \neq q$, $1 \leq p \leq m$ and $1 \leq q \leq m$. In other words, given stage–delay bounds, the delay of any stage at any moment could be any value as long as it is within the given bounds, disregarding the delays of adjacent stages. This is not true in real life. For example, if a combinational logic is partitioned into three stages, the delays in these three stages at any moment have some relationship which is defined by the implementation and function of this combinational logic. That is, they are not independent. One thing for sure is that the system, taking into account the stage–delay dependency, will have better performance (throughput bounds) than the same system which ignores the dependency.

2.> More logic delay terms: As shown in Chapter 5, if more logic delay terms are taken into account when doing the approximation, the result is closer to the exact bounds, with the price of adding more complexity. Trade–off analysis between complexity and accuracy is worthy.

3.> More basic control modules: In this thesis, we confine our performance investigation to several basic control modules, like FIFO, Fork, Join, Toggle/XOR, Arbiter/Call and Select/XOR, and the system consisting of some of these modules. To have more general applications and to facilitate design freedom, more basic control modules could be investigated. For example, Merge, True gate and False gate. Merge is similar to

Select except that input and output terminals are reversed. True (False) gate allows an input token to pass through only when control signal is true (False).

4.> Approximation–equivalent form: Sometimes it is easier to find performance bounds and any other equivalent input and output delays of a system if some parts of this system can be transformed into the other equivalent form. Since micropipelines communicate through a handshaking protocol, it looks as if most of the *exact* value for the equivalent input and output delays of any cross sections is a function of all of the stage–delays in a system. It would be hard to find an *exact* equivalent form of original circuit or system. For example, finding a form which is exactly equivalent to a Join circuit could be a challenging task. This is because to obtain an exact expression of the performance bounds of a Join circuit in terms of stage–delays (variable) is itself a challenge, not to mention obtaining its equivalent form. However, it could be easier to find its *approximation*–equivalent form, the form which has the equivalent result of approximation. For example, the approximation–equivalent form of a Join is shown in Figure 8.7. Some restrictions may be applied to the use of this equivalent form. Deriving properties may also help in constructing an approximation–equivalent form for a general circuit.

Of course, there are many other possible extensions to this research, besides the ones stated above. It is noted that, no matter what the research is, the features of simplicity and modularity are always desirable.

BIBLIOGRAPHY

[1] D. Pountain, "Computing Without Clocks," *BYTE*, pp.145–150, January 1993.

[2] C. Seitz, "System Timing," Chapter 7, *Introduction to VLSI Systems,* Addison–Wesley Publishing Company, 1980.

[3] Fumiyasu ASAI et al, "Self–Timed Clocking Design For A Data–Driven Microprocessor", IEICE Trans. Vol. E 74 No.11, pp.3757–3764, November 1991.

[4] S. Hauck, "Asynchronous Design Methodologies: An Overview," *Proceedings of the IEEE,* Vol. 83, No. 1, pp.69–93, January 1995.

[5] S.H Unger and C–J Tan, "Clocking Schemes for High–Speed Digital Systems," *IEEE Trans. on Computers*, Vol. C–35, No. 10, pp.880–895, October 1986.

[6] G.M. Jacobs and R.W. Brodersen, "Self–timed integrated circuits for digital signal processing applications," *VLSI Signal Processing, III*, R.Brodersen and H. Moscovitz, Eds. IEEE Press, pp.197–208, 1988.

[7] S.L. Lu, "Implementation of Micropipelines in Enable/Disable CMOS Differential Logic," *IEEE Trans. on VLSI Systems*, Vol. 3, No. 2, pp.338–341, June, 1995.

[8] M.E. Dean, D.L. Dill, and M. Horowitz, "Self–Timed Logic Using Current–Sensing Completion Detection(CSCD)," *Journal of VLSI Signal Processing*, 7, pp.7–16, 1994.

[9] I. E. Sutherland, "Micropipelines," *CACM*, Vol. 32, No. 6. pp.720–738, June 1989.

[10] J.C. Ebergen, *Translating Programs into Delay–Insensitive Circuits.* Centre for Mathematics and Computer Science, Amsterdam, CWI Tract 56, 1989.

[11] J.C. Ebergen, "A formal approach to designing delay–insensitive circuits," *Distributed Computing*, Vol. 5, No. 3, pp.107–119, July 1991.

[12] A.J. Martin, "Programming in VLSI: From communicating processes to delay–insensitive circuits," in *UT year of programming Institute on Concurrent Programming*, C.A.R. Hoare, Ed. Reading, MA:Addison–Wesley, pp.1–64, 1989.

[13] P. Kudva and V. Akella, "A Technique for Estimating Power in Asynchronous Circuits," *International Symposium on Advanced Research in Asynchronous Circuits and Systems,* Salt Lake City, Utah, USA, pp.166–175, November, 1994.

[14] J. A. Tierno and A. J. Martin, "Low–Energy Asynchronous Memory Design,"*International Symposium on Advanced Research in Asynchronous Circuits and Systems,* Salt Lake City, Utah, USA, pp.176–185, November, 1994.

[15] S.B Furber, P. Day, J.D. Garside, N.C. Paver and J.V. Woods, "AMULET1: A Micropipelined ARM," *proceedings of CompCon'94,* IEEE Computer Society Press, CompCon'94, San Francisco, March 1994.

[16] J. Tierno, A. Martin, D. Borkovic, and T. Lee, "A 100Mips GaAs asynchronous microprocessor," *IEEE Design and Test of computers*, Vol. 11, No. 2, pp.43–49, 1994.

[17] R.F. Sproull, I.E. Sutherland and C.E. Molnar, "Counterflow Pipeline Processor Architecture," *IEEE Design & Test of Computers*, pp.48–59, Fall 1994.

[18] C.M Chang and S.L. Lu, "Design of a Static MIMD Data Flow Processor Using Micropipelines," *IEEE Trans. on VLSI Systems*, Vol. 3, No. 3, pp.370–378, September, 1995.

[19] M.R. Greenstreet, *STARI: A Technique for High–Bandwidth Communications*, Ph.D thesis, Princeton University, January 1993.

[20] J.N. Seizovic, "Pipeline Synchronization," *International Symposium on Advanced Research in Asynchronous Circuits and Systems,* Salt Lake City, Utah, USA, pp.87–96, November, 1994.

[21] H. Hulgaard and S. Burns, "Bounded Delay Timing Analysis of a Class of CSP Program with Choice," *International Symposium on Advanced Research in Asynchronous Circuits and Systems,* Salt Lake City, Utah, USA, pp.2–11, November, 1994.

[22] H. Hulgaard, S. Burns, T. Amon, and G. Borriello, "An Algorithm for Exact Bounds on the Time Separation of Events in Concurrent Systems," *IEEE Transactions on Computers*, Vol. 44, No. 11, pp.1306–1317, November 1995.

[23] S. Burns, *Performance Analysis and Optimization of Asynchronous Circuits*, Ph.D Thesis, Computer Science Department, Caltech, 1991.

[24] T.Y. Wuu and S. Vrudhula, "Synthesis of Asynchronous Systems from Data Flow Specifications," *ISI Research Report, ISI/RR–93–366*, December 1993.

[25] C.V. Ramamoorthy and G. Ho," Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets," *IEEE Transactions on Software Engineering*, Vol. SE–6, No. 5, pp.440–449, September 1980.

[26] R.E. Muller, "Sequential Circuits," Chapter 10, In Switching Theory, Vol 2, Wiley, NY, 1965.

[27] T. E. Williams, "Performance of Iterative Computation in Self–Timed Rings," *Journal of VLSI Signal Processing*, 7, pp.17–31, 1994.

[28] L. Thiele, "On the analysis and optimization of selftimed processor arrays," *Integration, the VLSI journal* 12, pp.167–187, 1991.

[29] S.L. Lu and M. Ercegovac, "A Novel CMOS Implementation of Double–Edge–Triggered Flip–Flops", *IEEE Journal of Solid–State Circuits*, Vol.25, No.4, pp.1008–1010, August 1990.

[30] *Matlab User's Guide*, The Math Works, Inc., 1991.

[31] J.Y. Lin and D. Ionescu, "Asymptotic behavior of output feedback for a class of non–deterministic discrete event systems," *Int. Journal on Control*, Vol. 54, No. 4, pp.903–920, 1991.

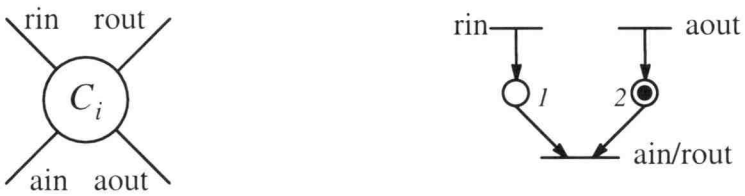
[32] G.J. Olsder, J.A.C. Resing, R.E. De Vries, M.S. Keane and G. Hooghiemstra, "Discrete Event Systems with Stochastic Processing Times," *IEEE Trans. on Automatic Control*, Vol. 35, No. 3, pp.299–302, March 1990.

[33] S.H Unger and C–J Tan, "Clocking Schemes for High–Speed Digital Systems," *IEEE Trans. on Computers*, Vol. C–35, No. 10, pp.880–895, October 1986.

[34] C.M. Chang and S.L. Lu, "Performance Issues on Micropipelines," *IEEE TCCA Newsletter*, Fall 1995.

APPENDICES

APPENDIX A: PETRI NET MODELS AND PROCESS NAMES FOR BASIC MODULES



CTSE process name: (rin,aout) c_element (ain)

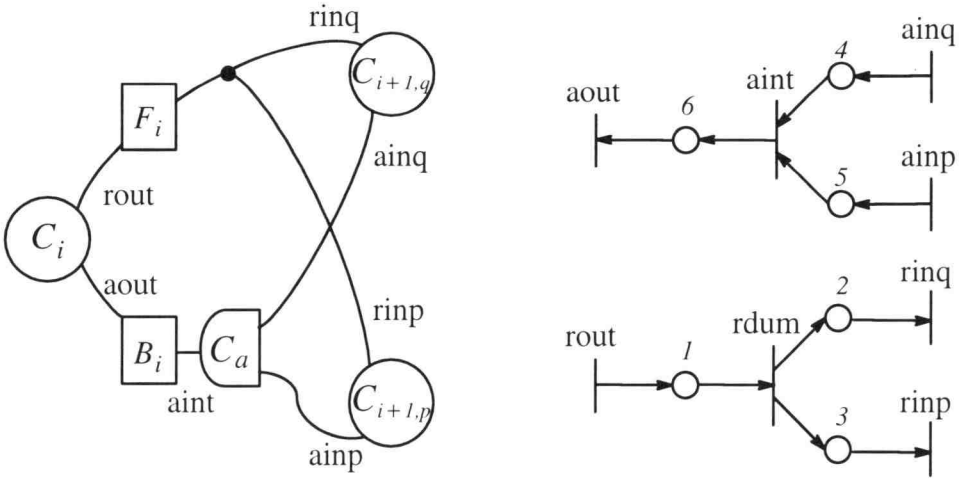
Figure A.1: Process of a C–element.

CTSE process name: (in) edge_no_token (out)

Figure A.2: Process of a delay path without initial token.

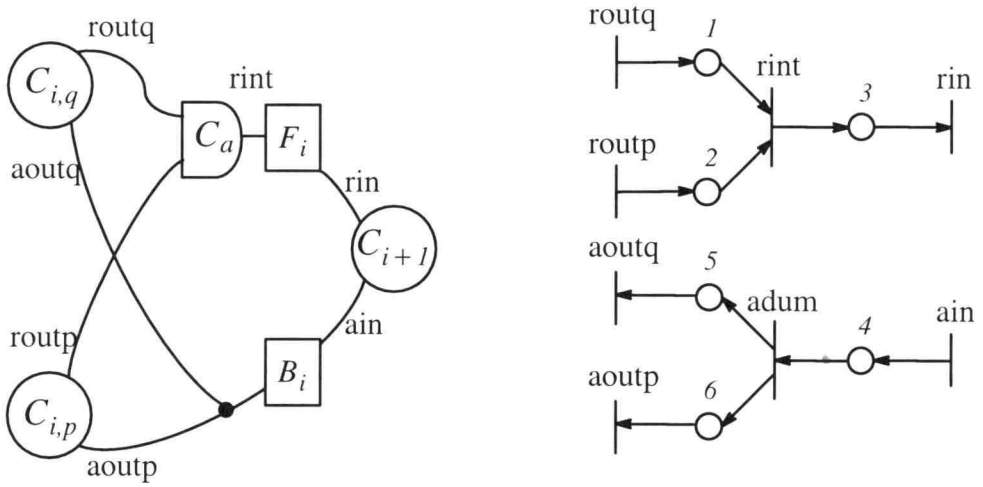
CTSE process name: (in) edge_with_token (out)

Figure A.3: Process of a delay path with initial token.



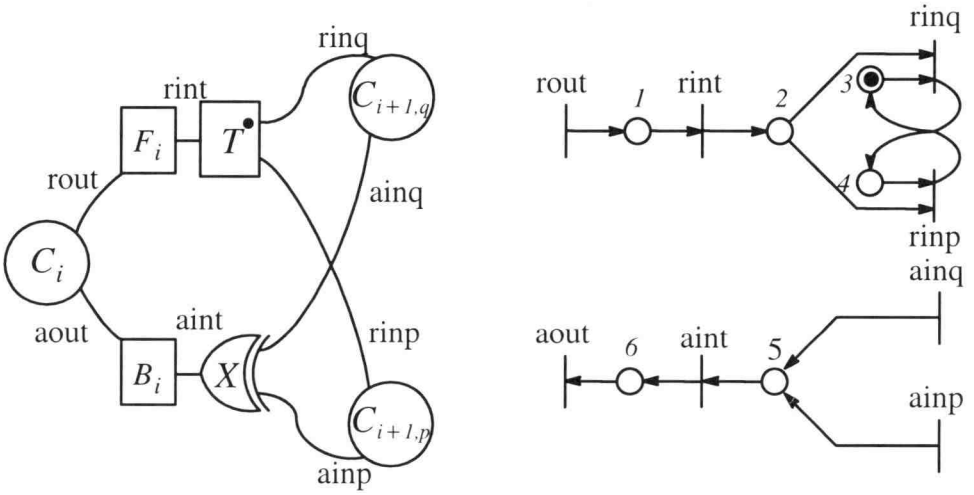
CTSE process name: (rout,ainq,ainp) fork (rinq,rinp,aout)

Figure A.4: Process of a Fork.



CTSE process name: (routq,routp,ain) join (rin,aoutq,aoutp)

Figure A.5: Process of a Join.



CTSE process name: (rout,ainq,ainp) toggle (rinq,rinp,aout)

Figure A.6: Process of a Toggle.

APPENDIX B: A CTSE CODE FOR A SYSTEM SHOWN IN FIGURE 8.1

```
// This is a CTSE code for the system shown in Figure 8.1
//unset timing;
set quiet;
let din_q_min=12;
let din_q_max=15;
let din_p_min=20;
let din_p_max=25;
let fi_q_min=4;
let fi_q_max=19;
let bi_q_min=3;
let bi_q_max=18;
let fi_p_min=19;
let fi_p_max=30;
let bi_p_min=6;
let bi_p_max=8;
let fi1_r_min=10;
let fi1_r_max=28;
let bi1_r_min=3;
let bi1_r_max=5;
let fi1_p_min=5;
let fi1_p_max=15;
let bi1_p_min=3;
let bi1_p_max=5;
let dout_t_min=29;
let dout_t_max=45;
let dout_u_min=11;
let dout_u_max=25;
let dout_p_min=2;
let dout_p_max=9;

let Celement=0;
let toggle_d=0;
let xor_d=0;
let dummy=0;

// define module c_element
define process (rin,aout) c_element (ain)
        {
        t(rin, aout, ain);
        s(p1,p2);

        fst(p1,ain);
        fst(p2,ain);

        fts(rin,p1);
        fts(aout,p2);

        // delay defined at each place
        delay(p1, Celement, Celement);
        delay(p2, Celement, Celement);
```

```
        // initial marking
        fts(root,p2);
        }

// define edge without initial token
define process (in) edge_no_token (out)
        {
        t(in, out);
        s(p1);

        fst(p1,out);
        fts(in,p1);
        }

// define edge with initial token
define process (in) edge_with_token (out)
        {
        t(in, out);
        s(p1);

        fst(p1,out);
        fts(in,p1);
        fts(root,p1);
        }

// define module fork
define process (rout,ainq,ainp) fork (rinq,rinp,aout)
        {
        t(rout,ainq,ainp,rinq,rinp,aout); // These transitions are for interface
        t(rdum,aint);  // These transitions are for internal use
        s(p1,p2,p3,p4,p5,p6);

        fst(p1,rdum);
        fst(p2,rinq);
        fst(p3,rinp);
        fst(p4,aint);
        fst(p5,aint);
        fst(p6,aout);

        fts(rout,p1);
        fts(rdum,p2);
        fts(rdum,p3);
        fts(ainq,p4);
        fts(ainp,p5);
        fts(aint,p6);

        delay(p2,dummy,dummy);
        delay(p3,dummy,dummy);
        delay(p4,Celement,Celement);
        delay(p5,Celement,Celement);
        }

// define module join
define process (routq,routp,ain) join (rin,aoutq,aoutp)
```

```
        {
        t(routq,routp,ain,rin,aoutq,aoutp);
        t(adum,rint);
        s(p1,p2,p3,p4,p5,p6);

        fst(p1,rint);
        fst(p2,rint);
        fst(p3,rin);
        fst(p4,adum);
        fst(p5,aoutq);
        fst(p6,aoutp);

        fts(routq,p1);
        fts(routp,p2);
        fts(rint,p3);
        fts(ain,p4);
        fts(adum,p5);
        fts(adum,p6);

        delay(p1,Celement,Celement);
        delay(p2,Celement,Celement);
        delay(p5,dummy,dummy);
        delay(p6,dummy,dummy);
        }

// define module toggle
define process (rout,ainq,ainp) toggle (rinq,rinp,aout)
        {
        t(rout,ainq,ainp,rinq,rinp,aout);
        t(rint,aint);
        s(p1,p2,p3,p4,p5,p6);

        fst(p1,rint);
        fst(p2,rinq);
        fst(p2,rinp);
        fst(p3,rinq);
        fst(p4,rinp);
        fst(p5,aint);
        fst(p6,aout);

        fts(rout,p1);
        fts(rint,p2);
        fts(rinp,p3);
        fts(rinq,p4);
        fts(ainq,p5);
        fts(ainp,p5);
        fts(aint,p6);

        delay(p2,toggle_d,toggle_d);
        delay(p3,dummy,dummy);
        delay(p4,dummy,dummy);
        delay(p5,xor_d,xor_d);

        fts(root,p3);
```

```
}
define process analysis_try_1
{
 t('riqi','aiqi','aiqo',
  'ripi','aipi','aipo',
  'r(i+1)ri','a(i+1)ri','a(i+1)ro',
  'r(i+1)si','a(i+1)si','a(i+1)so',
  'r(i+1)pi','a(i+1)pi','a(i+1)po',
  'r(i+2)ti','a(i+2)ti','a(i+2)to',
  'r(i+2)ui','a(i+2)ui','a(i+2)uo',
  'r(i+2)pi','a(i+2)pi','a(i+2)po');

instance eq ('aiqi') edge_with_token ('riqi');
delay (eq/p1, din_q_min, din_q_max);
instance ep ('aipi') edge_with_token ('ripi');
delay (ep/p1, din_p_min, din_p_max);
instance et ('a(i+2)ti') edge_no_token ('a(i+2)to');
delay (et/p1, dout_t_min, dout_t_max);
instance eu ('a(i+2)ui') edge_no_token ('a(i+2)uo');
delay (eu/p1, dout_u_min, dout_u_max);
instance ei2p ('a(i+2)pi') edge_no_token ('a(i+2)po');
delay (ei2p/p1, dout_p_min, dout_p_max);

instance cq ('riqi','aiqo') c_element ('aiqi');
instance cp ('ripi','aipo') c_element ('aipi');
instance cr ('r(i+1)ri','a(i+1)ro') c_element ('a(i+1)ri');
instance cs ('r(i+1)si','a(i+1)so') c_element ('a(i+1)si');
instance ci1p ('r(i+1)pi','a(i+1)po') c_element ('a(i+1)pi');
instance ct ('r(i+2)ti','a(i+2)to') c_element ('a(i+2)ti');
instance cu ('r(i+2)ui','a(i+2)uo') c_element ('a(i+2)ui');
instance ci2p ('r(i+2)pi','a(i+2)po') c_element ('a(i+2)pi');

instance toggle_m ('aiqi','a(i+1)ri','a(i+1)si') toggle ('r(i+1)ri','r(i+1)si','aiqo');
delay (toggle_m/p1, fi_q_min, fi_q_max);
delay (toggle_m/p6, bi_q_min, bi_q_max);

instance fifo_f_m ('aipi') edge_no_token ('r(i+1)pi');
delay (fifo_f_m/p1, fi_p_min, fi_p_max);
instance fifo_b_m ('a(i+1)pi') edge_no_token ('aipo');
delay (fifo_b_m/p1, bi_p_min, bi_p_max);

instance fork_m ('a(i+1)ri','a(i+2)ti','a(i+2)ui') fork ('r(i+2)ti','r(i+2)ui','a(i+1)ro');
delay (fork_m/p1, fi1_r_min, fi1_r_max);
delay (fork_m/p6, bi1_r_min, bi1_r_max);

instance join_m ('a(i+1)si','a(i+1)pi','a(i+2)pi') join ('r(i+2)pi','a(i+1)so','a(i+1)po');
delay (join_m/p3, fi1_p_min, fi1_p_max);
delay (join_m/p4, bi1_p_min, bi1_p_max);

}

// define separation time
define separation analysis_t_upper
        {
```

```
        process analysis_try_1;
        ref('a(i+2)ti');
        from(<'a(i+2)ti',3>);
        to(<'a(i+2)ti',2>);
        }


// define separation time
define separation analysis_u_upper
        {
        process analysis_try_1;
        ref('a(i+2)ui');
        from(<'a(i+2)ui',3>);
        to(<'a(i+2)ui',2>);
        }

// define separation time
define separation analysis_p_upper
        {
        process analysis_try_1;
        ref('a(i+2)pi');
        from(<'a(i+2)pi',3>);
        to(<'a(i+2)pi',2>);
        }

//lint(analysis_try_1);
//set po_alg = 2;
//set max_iter=1;

print analyze(analysis_t_upper);
print analyze(analysis_u_upper);
print analyze(analysis_p_upper);
```